Deformable Mesh Model for 3D Shape and Motion Estimation from Multi-Viewpoint Video

Shohei Nobuhara

Deformable Mesh Model for 3D Shape and Motion Estimation from Multi-Viewpoint Video

Shohei Nobuhara

Abstract

3D shape and motion estimation is an essential requisite for 3D video which records dynamic visual events in the real would as is. Although 3D video itself has various applications (e.g., 3D TV or 3D archive of intangible cultural assets, etc.), it also can be utilized as an input data of other applications such as 3D motion analysis.

In this thesis, we propose a framework to estimate 3D shape and motion from multi-viewpoint video using deformable mesh model. Our deformable mesh model is a kind of active contour model. It integrates multiple estimation cues such as photometric-consistency or silhouette boundary with local shape continuity consideration. Since each estimation cue represents a constraint which should be fulfilled if the mesh model represents the object shape and motion, we define forces each of which deforms the mesh model so as satisfy the corresponding constraint. So our deformable mesh model changes its shape so as to balance the forces working on vertices composing it.

With our deformable mesh model, we introduce two types of deformation. One is *intra-frame* deformation which estimates static object shape at a frame from multiple viewpoint images, and the other is *inter-frame* deformation which estimates the object shape and motion at two frames from multiple viewpoint video. In the intra-frame deformation, we utilize three constraints: photo-consistency, silhouette, and smoothness. These constraints defines *frame-and-skin* model to represent the object shape. That is, 1) the silhouette constraint defines a set of "frames" of the object, then 2) the smoothness constraint defines a rubber sheet skin covering the frames, and 3) the photometric constraint defines supporting points on the skin that have prominent textures. In the inter-frame deformation, we add constraints to model the object motion. We model the object motion as mixture of rigid motion and warping, and apply different deformation process to each vertex according to its property. We call this deformation process as *heterogeneous deformation*. To realize the heterogeneous deformation, we segment the

mesh model into rigid part and warping part by clustering the roughly estimated motion flow. Finally, we improve the heterogeneous deformation so that it can cope with time-varying global topology of the object. In the heterogeneous deformation process, we implicitly assumed a positive correlation between geodesic distance and Euclidean distance between vertices, and defined vertex forces with considering their geodesic local neighbors. However, changing of global topology brings global mesh collision and introduces a negative correlation between them which breaks down the geodesic proximity based force generation. We solve this problem by adding new Euclidean proximity based force which makes collided regions to repel each other so that they avoid global intersection which cannot deal with geodesic proximity based algorithm.

Acknowledgement

I would like to express my sincere appreciation to my adviser, Professor Takashi Matsuyama. He invited me to this interesting research when I was an undergraduate, and provided insightful advice and constant encouragement in these six years.

I would also like to thank other members of my thesis committee, Professor Michihiko Minoh and Professor Shigeru Eiho for taking the time to review my thesis and giving helpful comments.

Next, I would like to thank Professor Toshikazu Wada at Wakayama University and Associate Professor Akihiro Sugimoto at National Institute of Informatics. They supported my master's and bachelor's thesis when they were in Kyoto University.

I am deeply indebted to Visiting Professor Kazuhiko Sumi, Associate Professor Atsuto Maki, Assistant Professor Hitoshi Habe, Assistant Professor Hiroaki Kawashima, and all the members of Matsuyama laboratory, especially my colleagues Xiaojun Wu and Takeshi Takai.

My thanks also go to Genichi Imamura and Akihiro Hatanaka. They are the President and Director of Four-Dimensional Data, Inc., and gave me the opportunity to cover living expenses.

Finally, I have to thank my family for everything they have done for me.

Contents

Chapter 1

Introduction

1.1 Background

The meaning of the phrase *"shot of a person"* has been changed by the progress of technology. In the 19th century, "a shot" meant a photograph. A lot of innovative technologies, e.g., optical lens and films, had been developed at this time. Although it was a still image, we can imagine how it would been surprising if we had never seen a photograph. In the 20th century, "take a shot" acquired another meaning – a video. Today, we cannot live without these visual media because we make full use of visual information in our everyday lives.

In this early 21st century, we are now about to launch a new visual media: *3D video*[MTG97]. 3D video records dynamic visual events in the real would as is. That is, it records time-varying 3D object information while conventional video records 2D information. The applications of 3D video cover a wide spectrum of human activities: entertainment (3D TV, game), education (3D picture book of animals), sports (3D playback, coaching), and culture (3D archive of intangible cultural assets). We believe that 3D video makes unimaginable impact on human society as 2D photograph and video had done in 19th and 20th centuries.

To realize 3D video, we have to develop a lot of technologies for capturing, storage, editing, transmission, and visualization as those developed for 2D conventional video. In this thesis, we focus on an algorithm for 3D video capturing. Capturing of 3D object has been an attractive research topic in computer vision. Most of the proposed 3D recording systems use multi-viewpoint cameras circumnavigating a target object [MTG97] [KRN97] [WWTM00] [BD00] [CKBH00] [MWTN04]. Figure 1.1 shows an example of multi-viewpoint input images and estimated object shape.

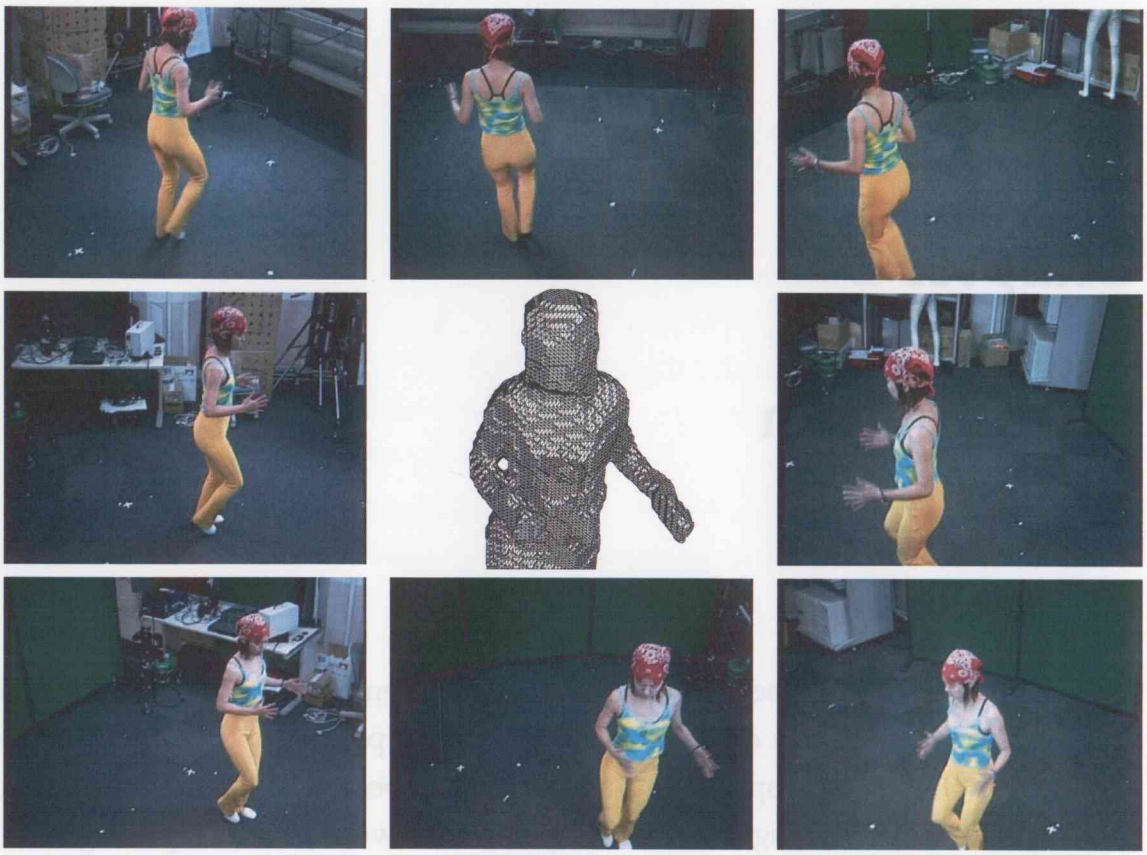


Figure 1.1: Input images and visual hull

While most of the systems focused on the estimation of 3D shape or motion individually, the problem we consider in this thesis is how we can estimate the 3D object shape and motion simultaneously from captured multi-viewpoint video. The reason why we estimate not only the 3D shape but also the 3D motion is that if we can obtain the object motion as dense correspondences between two frames, it can be used for inter-frame data compression[BSM+03][IR03] and motion analysis. It is discussed in Section 1.3.1.

1.2 3D Shape and Motion Estimation: Problem Specification

Before we introduce our deformable mesh model in Section 1.3.1, we review several 3D shape and motion estimation algorithms proposed so far.

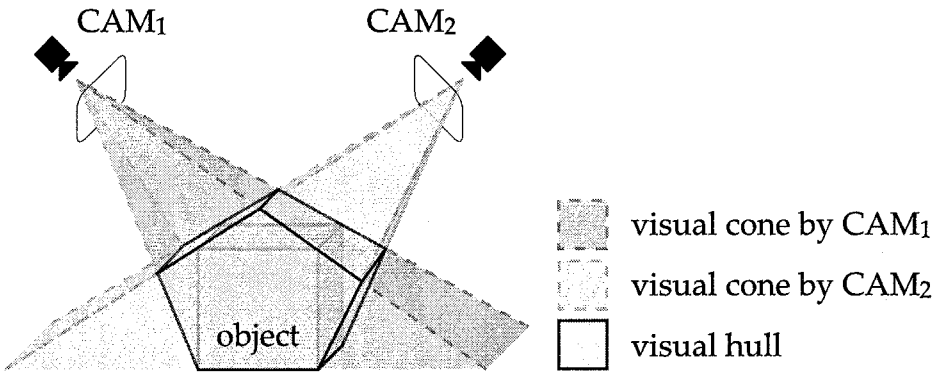


Figure 1.2: Visual hull of the object

1.2.1 3D Shape Estimation

In this section, we first distinguish *2.5D* shape reconstruction and *full 3D* shape reconstruction, both of which are sometimes called "3D shape reconstruction." 2.5D shape (also called as depth, range, or Z- image/map) is a half, one-sided shape of a 3D object typically given by stereo method, and back side of the object is unknown. Full 3D shape, on the other hand, is the whole shape of the object, and in this thesis, we focus on how we can obtain it. In this context, 2.5D shape reconstruction method can be turned into full 3D method by combining several 2.5D shapes of various sides of the object into single full 3D shape. This process is called *registration*[WSI98]. Note that the term "3D shape" means full 3D shape unless otherwise stated hereafter.

To obtain 3D shape, we have to observe an object from different viewpoints. That is, we need multi-viewpoint images of the object. In general, we have two choices to realize multi-viewpoint observation of the object. One is a multiple camera system in which cameras are arranged so as to circumnavigate an object [WWTM00]. We can refer to this as *spatial* multi-viewpoint system. The other one is a *temporal* multi-viewpoint system. It uses single camera and rotate it around the object to capture images from different viewpoints[WC01]. Obviously, temporal multi-viewpoint system asks the object to be fixed while it takes multi-viewpoint images. Although it is a reasonable requirement if we capture a solid object (e.g., building or statue), it is not suitable for capturing time-varying 3D object. So we assume that we use spatial multi-viewpoint system to capture a 3D object in this thesis.

From each view of multi-viewpoint system, we can obtain several kind of in-

formation as input data. Typically, 2.5D shape estimation methods utilize object texture[KRN97], shading[WUM95], or motion[VBR+99] to generate depth map. On the other hand, full 3D shape estimation methods utilize object silhouette[Lau95], texture[KS99], or contour[SF95]. That is, 2D information at each viewpoint. Hence, 3D shape estimation methods can be categorized into two types based on what each viewpoint gives. One is two-step method which first estimate 2.5D shapes at each viewpoint and then integrates them into single full 3D shape. The other one is one-step method which estimates full 3D shape directly from 2D information at each viewpoint.

These methods have own advantages and disadvantages based on the estimation cues which they employ. For example, texture-matching based stereo[KRN97] or space carving[KS99][SD99] can estimate arbitrary visible surfaces if it has prominent texture, but we cannot expect that the object surface has prominent textures all over it. On the other hand, volume intersection method[Lau95] estimates full 3D shape from 2D object silhouettes at each viewpoint. In general, 2D silhouette estimation is stable than texture-matching, but volume intersection produces the visual hull of the object. Here, the visual hull is the intersection of visual cones each of which is generated by projecting the observed object silhouette from camera center(Figure 1.2). It ensures that the object is encaged in it by its definition, but it cannot represent such portions of object shape that cannot be observed from viewpoints as a part of silhouette contour. Roughly speaking, the visual hull cannot represent concave portions. So recent methods propose frameworks which combine multiple estimation cues to accomplish more accuracy and stability. For example, Fua[FL94][Fua95][FL95] represented object shape by a 2.5D triangular mesh model and deformed it based on photometric stereo and silhouette constraint. Cross[CZ00] and Esteban[ES02] carved a visual hull, a set of voxels given by silhouettes, using photometric properties.

1.2.2 3D Shape and Motion Estimation

Similarly to full 3D shape estimation described above, we need spatial multi-viewpoint video to estimate 3D shape and motion. Note here that there are some algorithms which deal with multi-viewpoint video and estimate 3D object shapes at each frame, but not estimate 3D motions between them. For example, Goldlüecke[GM04] proposed a 4D space-carving algorithm which estimates

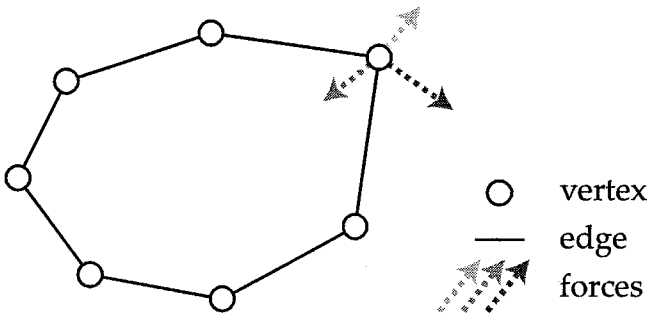


Figure 1.3: Active contour model

time-varying object shape as an isosurface in 4D (3D geometry + 1D time) space.

For object shape and motion estimation, there are two types of algorithm proposed so far. One is a method which requires object model given a priori. For example, Heap[HH96] proposed an algorithm which tracks human hand using hexagonal mesh model. Plänkers[PF03] employed a soft object model given a priori. They first track the object, and then refine its shape based on photometric and silhouette constraints. The other one is a method which only uses input images. Vedula[VBSK00] proposed a 6D space carving algorithm which carves 6D correspondence space based on "photo-consistency," and generates object shapes and per-voxel correspondences between two frames.

Since the formers require additional object model, it is not suitable for general 3D shape and motion estimation. On the other hand, the latter algorithm totally depends on the textures of the object surface as like as 2.5D stereo method, but it is difficult to expect that every surface regions have prominent texture.

Hence, as discussed for 3D shape estimation algorithms, it is required to integrate multiple estimation cues for 3D shape and motion estimation so that we can accomplish both accuracy and stability.

1.3 Deformable Mesh Model

In this thesis, we introduce "deformable mesh model" as basic computation framework. This is a kind of 3D active contour model[KWT88][SHIR95] which represents the 3D object shape. As reviewed in the previous section, mesh model (active contour model) is a reasonable scheme to

- integrate several cues for accuracy, and

- realize shape continuity for stability.

Integration of multiple cues has following meanings.

Least commitment principle If one cue suggests an optimal state, we cannot determine whether it is actually an optima or a fake cause by noise. In this case, we have no choice but to determine based on a certain threshold if we use the stereo or other single-cue algorithms. However, with active contour model, such an uncertain cue can avoid decision (i.e., thresholding) until others be in consensus.

Mutual compensation Each cues have own advantages and disadvantages. For example, silhouette-based volume intersection method is stable, but cannot reconstruct accurate 3D object shape; its output represents just the visual hull of the object and concave portions of the object cannot be reconstructed. In contrast, texture-based stereo method can reconstruct any kind of visible surfaces, but it is difficult to obtain dense and accurate stereo matching. Active contour model can combine both cues. If an object surface has a prominent texture, its position is determined accurately by mean of the stereo method. Otherwise, its position is smoothly interpolated according to its neighbors whose positions are determined by texture or silhouette.

Detailed situation analysis Active contour model realize better integration rather than just mixing several evaluated values. It can adaptively control the weightings of each cues based on the analysis between cues reflecting the situation of the object shape and motion.

In general, active contour model consists of vertices and edges connecting each vertex(Figure 1.3), and deforms its shape so as to balance the forces working at each vertex. Integration of several cues is realized as follows:

1. Suppose we have several cues. Each of them represents a constraint which should be fulfilled if the active contour model represents the object.

2. For each estimation cue, let us define a force working at each vertex. Each of forces moves each vertex so as to satisfy the corresponding constraint.

3. By moving each vertex according to the combined forces, we can expect that each vertex will be placed where the multiple constraints are fulfilled.

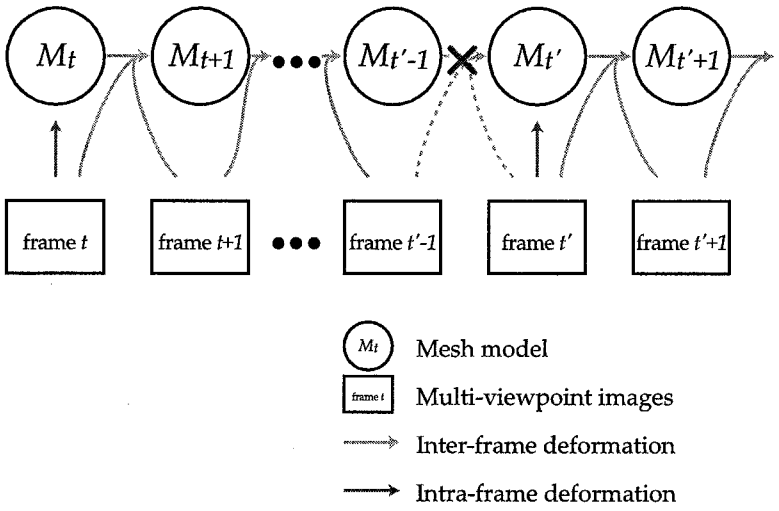


Figure 1.4: 3D video capturing using deformable mesh model

Here, each of constraints corresponds to a method which utilize single cue, e.g., stereo method or shape from silhouette.

On the other hand, shape continuity for stability is realized by adding new force which constrains the positions of vertices by their neighbors. This can be considered as "smoothness constraint." While it does not estimate the object shape itself, it can be also integrated as same as other cues.

1.3.1 3D Shape and Motion Estimation Using Deformable Mesh Model

In this thesis, we introduce two types of deformation using deformable mesh model: the intra-frame deformation and the inter-frame deformation. The former is an algorithm to estimate the still object shape at a single frame, and the latter is an algorithm to estimate the object shape and motion from frame t to $t+1$. The inter-frame deformation uses the object shape at t as its initial shape, and deforms it so as to represent the object shape at $t+1$. Here, since we deform from t to $t+1$, we know per-vertex correspondences as deformation loci, so we can regard it as dense, per-vertex object motion.

To realize 3D video, we use these two deformations as illustrated in Figure 1.4:

Step 1. Estimate the object shape at frame t by the intra-frame deformation. We denote this mesh model by M_t.

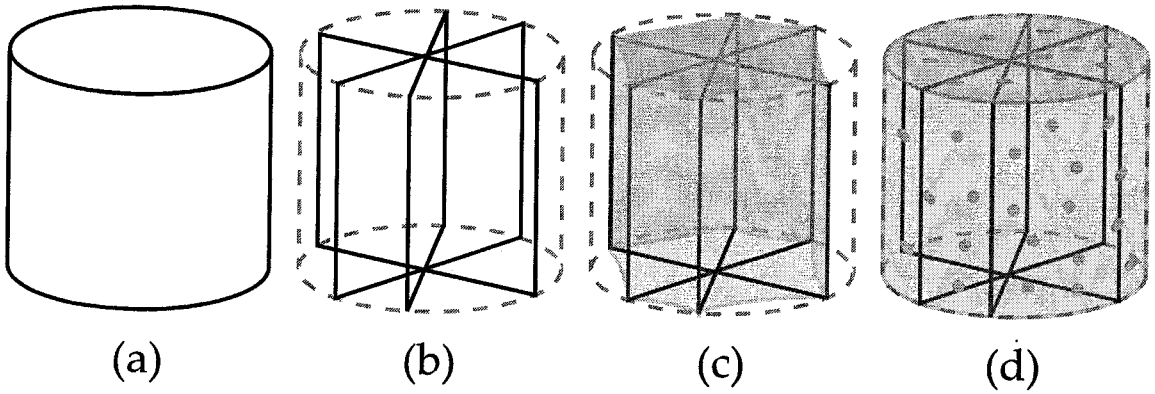


Figure 1.5: Frame and skin model.
(a) original object, (b) frames, (c) rubber skin over the frames, (d) supporting points on the skin.

Step 2. Estimate the object shape and motion at frame $t+1$ by the inter-frame deformation. Here, we use M_t as the initial shape, and obtain M_{t+1}.

Step 3. Repeat Step 2. and obtain M_{t+2} from M_{t+1}, M_{t+3} from M_{t+2}, and so on.

Step 4. Suppose the result of the inter-frame deformation $M_{t'}$ at frame t' cannot achieve a user-defined shape quality. In this case, estimate $M_{t'}$ by the intra-frame deformation, then go back to Step 2. and estimate $M_{t'+1}$ from $M_{t'}$.

This deformation process can be explained on the analogy of 2D video compression. We can apply inter-frame compression for mesh data obtained by the inter-frame deformation[BSM+03][IR03], and for error-recovery, we can insert key-frames by the intra-frame deformation.

1.4 Outline

As described above, we introduce two types of deformation.

In Chapter 2, we introduce the intra-frame deformation algorithm to estimate the static 3D object shape. In this chapter, we employ three constraints to estimate 3D shape: photo-consistency, silhouette, and smoothness. These three constraints define a *frame and skin model* to represent the 3D object shape:

1. Suppose we want to model the object in Figure 1.5 (a).

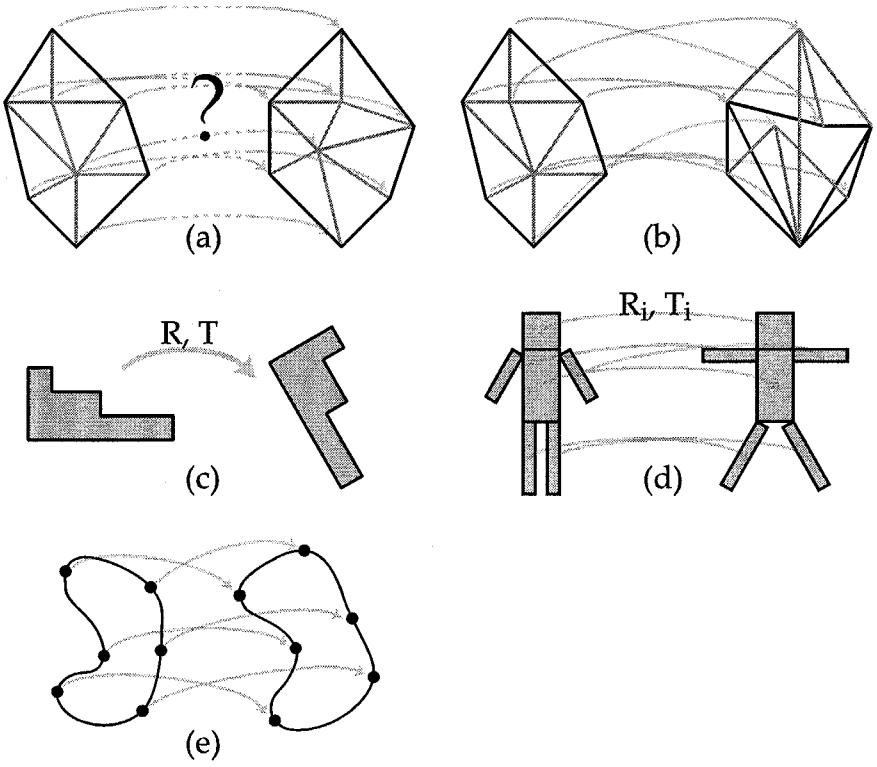


Figure 1.6: Object motion models

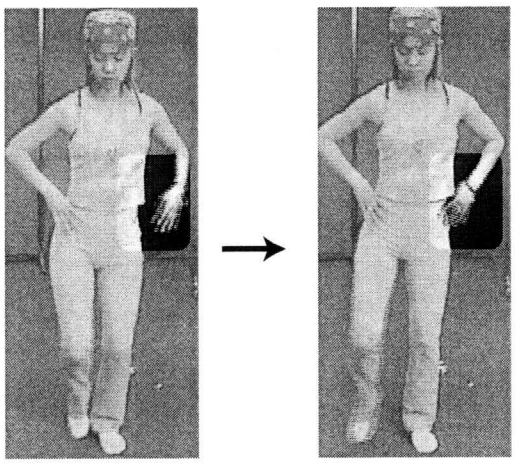
2. First, the silhouette constraint defines a set of frames of the object (Figure 1.5 (b)).

3. Then the smoothness constraint defines a rubber sheet skin covering the frames (Figure 1.5 (c)).

4. Finally, the photo-consistency constraint defines supporting points on the skin that have prominent textures (Figure 1.5 (d)).

and they act so as to fit the mesh to the real object shape based on each estimation cues.

In Chapter 3, 4 and 5, we introduce the inter-frame deformation algorithm to estimate the object shape and motion simultaneously. In Chapter 2, we have introduced the shape model of the object defined by three estimation constraints, so we add new constraints to model the object motion. First, we represent the object motion as translations of vertices composing the deformable mesh model, and the problem is how we can estimate these translation vectors of vertices (Figure 1.6(a)). Without any constraint on the object motion, per-vertex translations may represent a physically-unreasonable object motion as illustrated in Figure 1.6(b).



frame t frame t+1

Figure 1.7: An example of object with time-varying global topology

In this thesis, we categorize the object motion into following four types: rigid motion, articulated motion, warping, and mixture of them [YS03a]. In rigid motion, the translation vectors of vertices are governed by only one rotation and translation (Figure 1.6(c)). In articulated motion, the vertices are segmented into several parts, and each part moves rigidly (Figure 1.6(d)). In warping, the deformable mesh model changes its shape freely from a global point of view, but each vertex keeps its local arrangement between its neighbors (Figure 1.6(e)). In the mixture model, the vertices are clustered into rigid or warping portions, and each vertex moves according to this categorization. Obviously, rigid or articulated motion lacks an ability to represent an object motion like a human accurately, but it is stable to estimate their motion parameters. On the other hand, warping has an ability to represent accurate object motion, but it is hard to estimate translation vectors for vertices without prominent textures, that is, unidentifiable surface regions. So we employ the mixture model, and introduce *heterogeneous deformation model* which moves vertices according to their 1) photometric properties (i.e., if they have prominent textures or not) and 2) physical properties (i.e., if they belong to a rigid part or warping part).

Chapter 3, 4 and 5 are organized as follows. In Chapter 3, we introduce basic inter-frame deformation by modeling the object motion as warping. In this chapter, we assume that the object surface has prominent textures all over it. In Chapter 4, we introduce heterogeneous deformation model which models the object motion as mixture of rigid motion and warping, and changes per-vertex

deformation processes according to their photometric and physical properties. In Chapter 5, we extend the heterogeneous deformation process so that it can cope with time-varying global topology (Figure 1.7). In the heterogeneous deformation process, we implicitly assumed a positive correlation between geodesic distance and Euclidean distance between vertices, and defined vertex forces with considering their geodesic local neighbors. However, changing of global topology brings global mesh collision and introduces a negative correlation between them which breaks down the geodesic proximity based force generation. We solve this problem by adding new Euclidean proximity based force which makes collided regions to repel each other so that they avoid global intersection which cannot deal with geodesic proximity based algorithm.

Finally, we conclude this thesis in Chapter 6 with discussions and possible future research directions.

Chapter 2

Deformable Mesh Model for Static 3D Shape Estimation

In this chapter, we show our deformable mesh model which reconstruct a static object shape from multi-viewpoint images. This is a method for key-frame estimation of 3D video (in Section 1.3.1), and the basic model to be augmented for the inter-frame deformation in the next chapter.

As described in Chapter 1.2, 3D shape reconstruction methods from single reconstruction cue have own advantages and disadvantages. For example, the volume intersection method is stable, but cannot reconstruct accurate 3D object shape; its output represents just the visual hull of the object and concave portions of the object cannot be reconstructed. In contrast, the stereo method can reconstruct any kind of visible surfaces, but it is difficult to obtain dense and accurate stereo matching. Moreover, since the stereo analysis merely generates a 2.5D shape, multi-view stereo data should be integrated to reconstruct the full 3D object shape.

To accomplish more stability and accuracy, recent methods proposed frameworks to combine multiple reconstruction cues. For example, Fua[FL94] represented object shape by a 2.5D triangular mesh model and deformed it based on photometric stereo and silhouette constraint. Cross[CZ00] carved a visual hull, a set of voxels given by silhouettes, using photometric properties.

In this chapter, we show a mesh-deformation method for full 3D shape reconstruction[IS02] which can integrate multiple reconstruction cues. This is also the basic scheme to be augmented for the dynamic 3D shape reconstruction in Chapter 3. First, we describe the problem now considered in Section 2.1, and our approach to solve the problem in Section 2.2. Then we show reconstruction

cues to be used in Section 2.3, and how we design our deformable mesh model (Section 2.4 and 2.5). Several experimental results and discussions are given in Section 2.6.

2.1 Problem Description

The problem we consider in this chapter is static 3D shape estimation from multi-viewpoint images. Here, we assume the estimation target as follows:

- it has arbitrary shape with smooth and continuous surface, and
- it has known reflectance property.

We use Lambertian reflectance model: observed color of the object surface does not depend on the camera position. This is because that we assumed arbitrary shape, and simultaneous estimation of unknown shape and reflectance property without strict shape model is very difficult[WUM95] since most of reflectance model depends on the surface normal which is to be estimated as well.

Next, we define the input, multi-viewpoint images as follows:

- object images captured by *calibrated* cameras circumnavigating the object and
- object silhouettes segmented from the object images,

where *calibrated* means that all the intrinsic and extrinsic parameters of cameras are known. Note that we can consider that observed color of the object surface is consistent between cameras since we assumed Lambertian surface and calibrated cameras.

2.2 Approach

As described in Section 1.3, we use deformable mesh model to integrate multiple estimation cues so as to compensate their disadvantages each other. Figure 2.1 shows our deformable mesh model. It consists of the following elements to represent the object shape:

- vertices,
- edges each of which connects two vertices, and

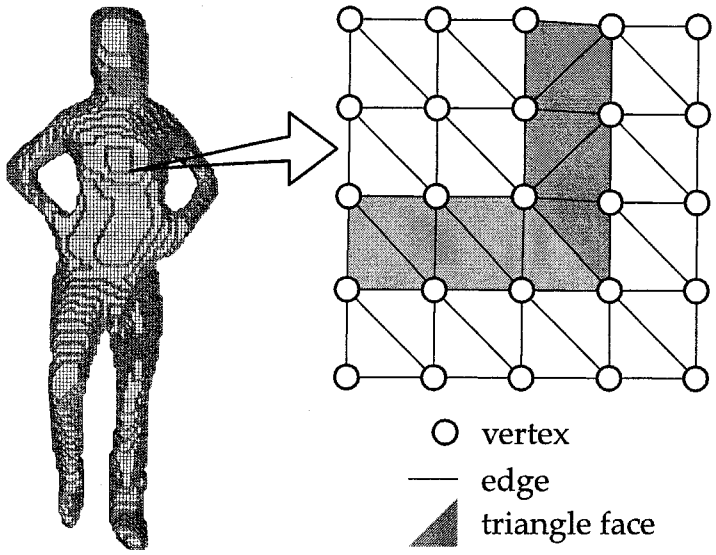
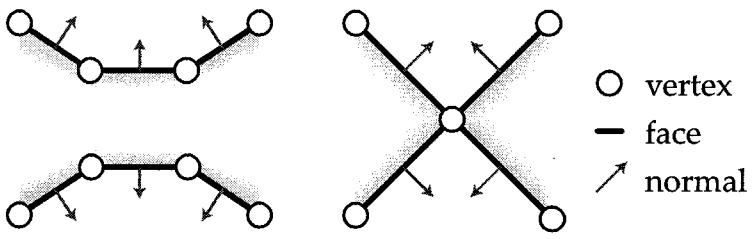


Figure 2.1: Deformable mesh model



(a) tow-manifold (b) four-manifold

Figure 2.2: Two-manifold and four-manifold mesh models

- triangle faces defined by three edges.

As illustrated in the right of Figure 2.1, each vertex has own connectivities between their neighbors, we refer to this per-vertex connectivities as *local topology* of the mesh model. On the other hand, the left of Figure 2.1 illustrates entire shape of the mesh model which has two holes. In other words, this is a genus-2 mesh. We refer to this genus number as *global topology* of the mesh model. Note here we assume that our mesh model has no isolated vertices, edges, faces, and no local holes. That is, our mesh model is a closed, two-manifold discrete triangle mesh model in which all edges are shared by exactly two faces (Figure 2.2).

With this deformable mesh model, we employ stereo-matching and silhouette as constraints which should be satisfied by the real shape of object by introducing

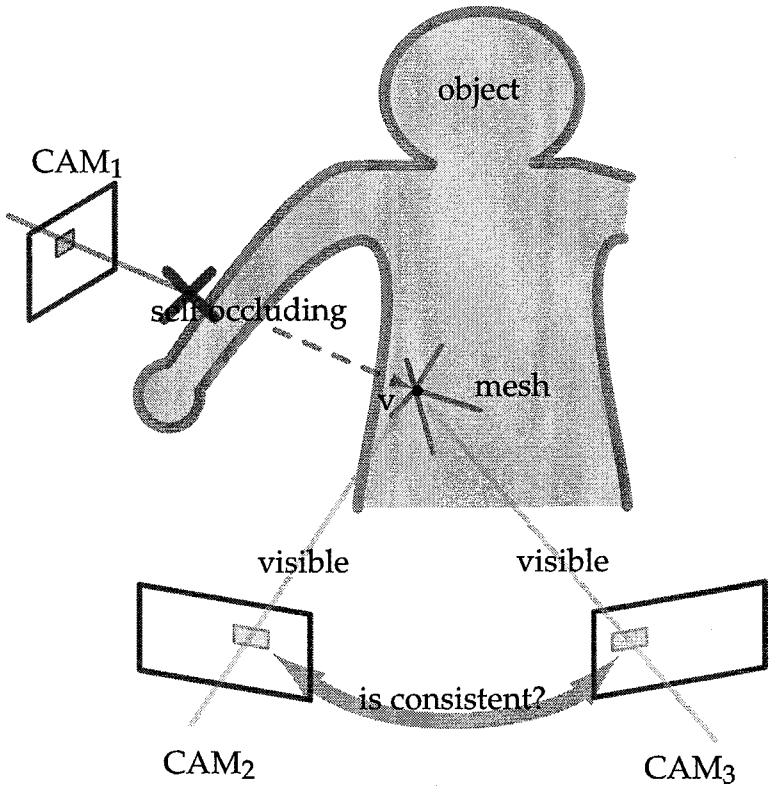


Figure 2.3: Photometric consistency and visibility

forces derived from these constraints.

2.3 Constraints for Estimation

2.3.1 Photometric Constraint

> *A patch in the mesh model should be placed so that its texture, which is computed by projecting the patch onto a captured image of visible viewpoint, should be consistent irrespectively of onto which image it is projected (Figure 2.3).*

Here, *visible* viewpoints for a vertex is a set of viewpoints from which the vertex can be observed. In other words, the vertex is not self-occluded by another part of the mesh model. We henceforth denote the set of those cameras that can observe a vertex v by C_v. Note again that C_v depends on not only the position of the vertex v and viewpoints, but the global shape of the mesh model, i.e., the positions of other vertices.

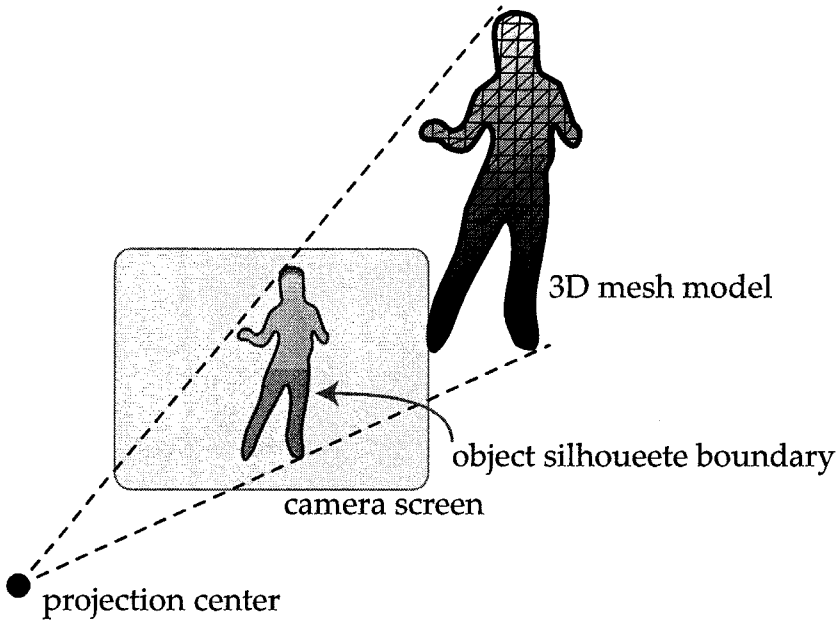


Figure 2.4: Silhouette constraint

This constraint represents a multi-baseline stereo shape estimation with self-occlusion consideration.

2.3.2 Silhouette Constraint

> *When the mesh model is projected onto an image plane, its 2D silhouette boundary should be coincide with the observed object silhouette boundary on that image plane (Figure 2.4). Moreover, vertices corresponding to 2D silhouette should be placed continuously on the 3D object surface.*

Figure 2.5 shows relationship between 2D silhouette on a viewpoint and 3D object shape: each point on the 2D silhouette boundary has a corresponding 3D point on the object 3D surface, called *contour generator*. As we assumed the object surface to be smooth and continuous, we can expect that contour generators on the object surface are placed continuously as well. Strictly speaking, particular object shape and camera arrangement may map multiple contour generators to single 2D silhouette boundary point (Figure 2.6). However, by supposing that cameras are in general position and object surface is smooth, we assume that only one vertex corresponds to a 2D silhouette boundary point.

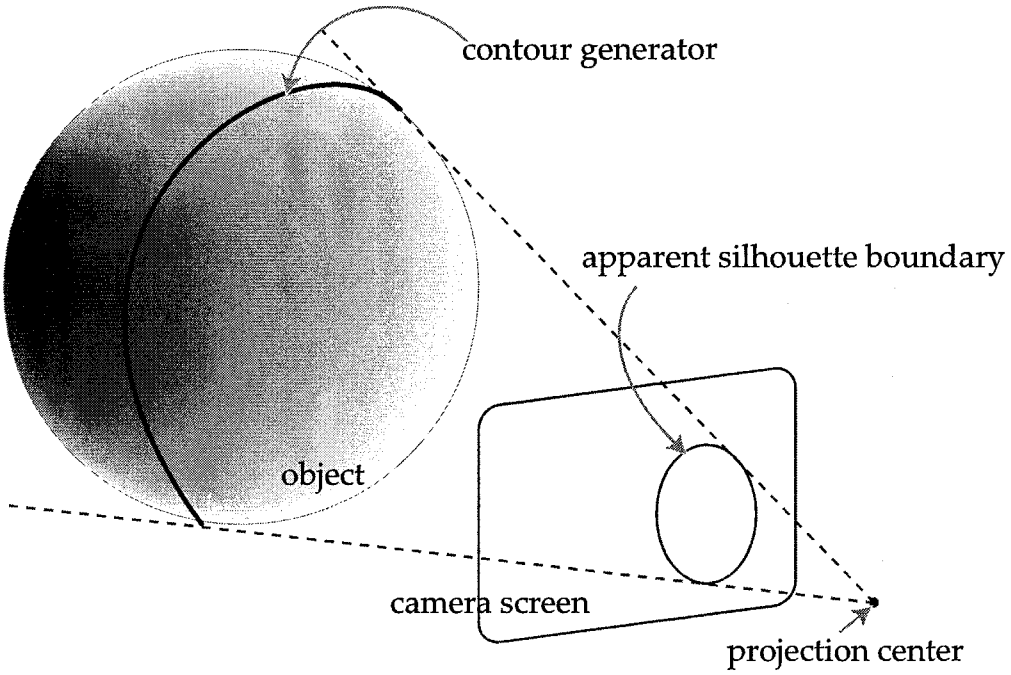


Figure 2.5: Contour generator

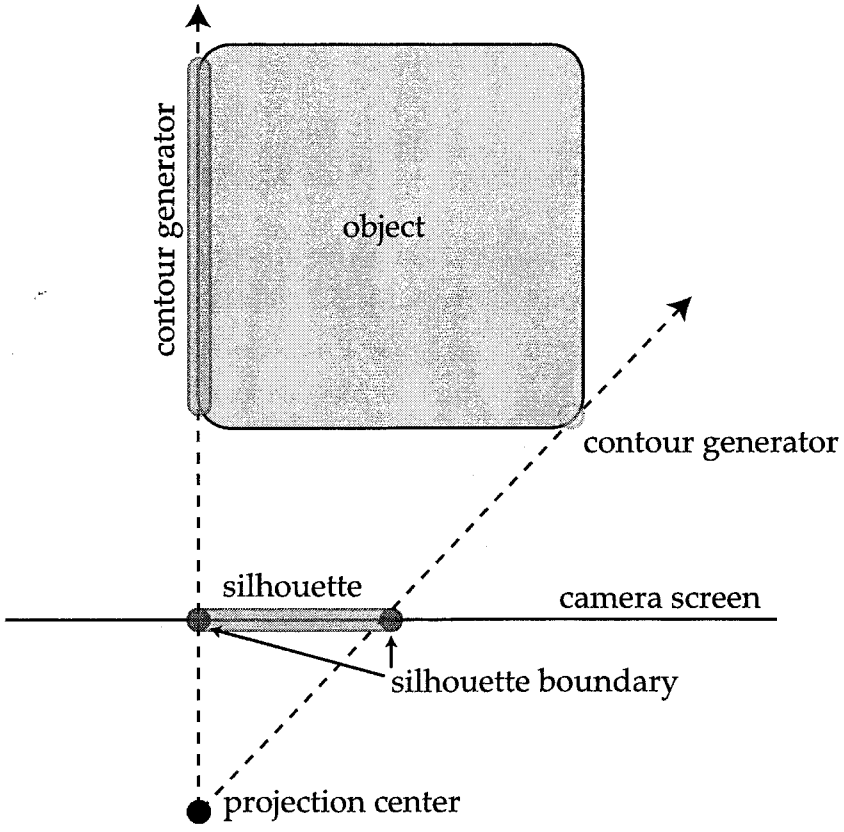


Figure 2.6: Contour generator and camera arrangement

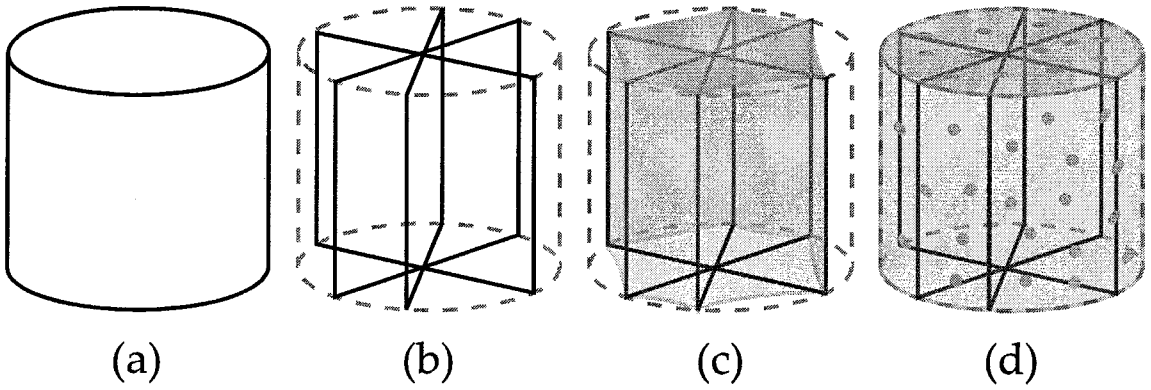


Figure 2.7: Frame and skin model.
(a) original object, (b) frames, (c) rubber skin over the frames, (d) supporting points on the skin.

This constraint makes the mesh model to be tangent to the visual cone defined by a viewpoint.

2.3.3 Smoothness Constraint

The 3D mesh should be locally smooth and should not intersect with itself.

We use a two-manifold surface model to represent a mesh model (see below), and assume that the object surface can be represented by smooth and continuous connectivity of vertices, that is, without local protrusions, dents, holes, and self-intersections.

2.3.4 Frame-and-Skin Model

Three constraints described above define a *frame and skin model* to represent 3D object shape:

1. Suppose we want to model the object in Figure 2.7 (a).
2. First, the silhouette constraint defines a set of frames of the object (Figure 2.7 (b)).
3. Then the smoothness constraint defines a rubber sheet skin covering the frames (Figure 2.7 (c)).

4. Finally, the photometric constraint defines supporting points on the skin that have prominent textures (Figure 2.7 (d)).

and they act so as to fit the mesh to the real object shape based on each estimation cues. Note that frames, skins, and supporting points are described as if they are determined one by one, but they are estimated simultaneously.

2.4 Design of Deformable Mesh Model

2.4.1 Representation

Shape

As described above, we use two-manifold surface model to represent an object shape. We define a surface not functionally (e.g., Bezier or NURBS surface), but as a set of triangle patches called triangle mesh, so our mesh model consists of vertices, edges, and faces with Winged-Edge or Half-Edge structure as usual mesh model. This is because:

- We need to project a surface onto camera planes and compare the textures of projected regions to implement the photometric constraint. It is difficult, however, to project a functional surface onto a plane as is, and in many cases, functional surfaces will sub-divided into small planar triangles to project.

- In this thesis, we use the visual hull of the object as the initial shape of the deformation process (described below). Typically, the visual hull is obtained in volumetric representation, e.g., set of voxels, but it is not so easy to convert a volumetric representation into functional surface representation while conversion into polygonal surface has well-known methods[LC87][BG93][KKI99].

Constraints

To realize the shape deformation like SNAKES[KWT88], we can use either energy function based or force based methods. As described above, we employed a force based method. This is firstly, from a computational point of view, because we have too many vertices (for example, the mesh model shown in Figure 1.1 has

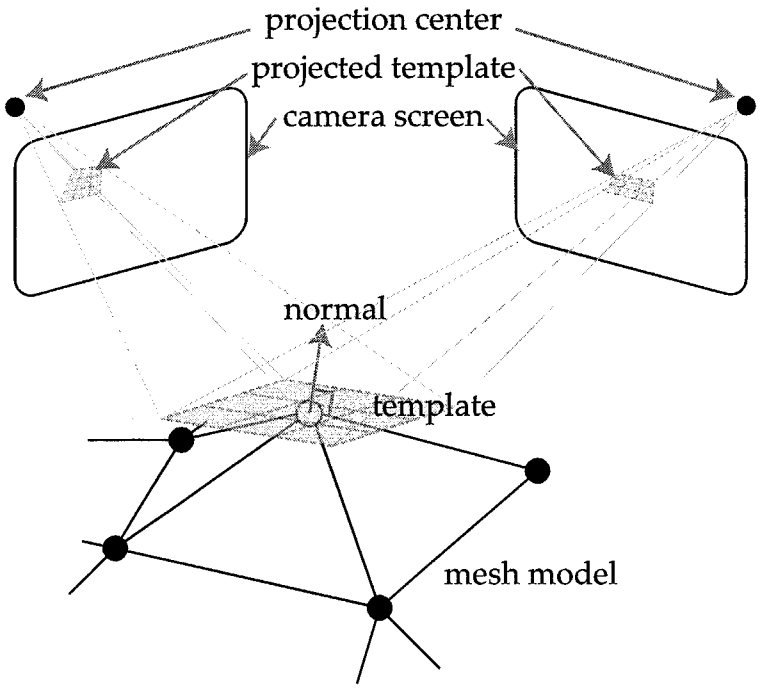


Figure 2.8: Template matching

about 12,000 vertices) to solve energy function, and secondly, from an analytical point of view, because the silhouette constraint cannot be represented as any analytical energy function (see below).

2.4.2 Forces at each Vertex

Here, we give how we design each constraints as forces working on each vertices. The constraints we mentioned above should be satisfied if the deformable mesh represents the real shape of the object. So we define the forces to move a vertex so as to satisfy each constraints.

Photometric Force

First, we define the photometric force $\boldsymbol{F}_e(v)$ to deform the mesh to satisfy the photometric constraint.

$$\boldsymbol{F}_e(v) \equiv \begin{cases} \nabla E_e(\boldsymbol{q}_v), & \text{if} \quad N(C_v) \geq 2, \\ 0, & \text{otherwise}, \end{cases} \tag{2.1}$$

where $N(C_v)$ denotes the number of cameras in C_v, $E_e(\boldsymbol{q}_v)$ the correlation of textures to be mapped around v (Figure 2.3):

$$E_e(\boldsymbol{q}_v) \equiv \frac{1}{N(C_v)-1} \sum_{c \in C_v \setminus c_m} \mathrm{NCC}\,(c, c_m)\,, \tag{2.2}$$

where c_m denotes the most facing camera in C_v, c a camera in C_v except c_m, $\mathrm{NCC}\,(c, c_m)$ the normalized cross correlation function between c and c_m given by:

$$\mathrm{NCC}\,(c, c_m) \equiv \frac{\iint_{w_v} \left(p_{w_v,c}\,(x,y) - \overline{p_{w_v,c}}\right)\left(p_{w_v,c_m}\,(x,y) - \overline{p_{w_v,c_m}}\right) dxdy}{\sqrt{\iint_{w_v} \left(p_{w_v,c}\,(x,y) - \overline{p_{w_v,c}}\right)^2 dxdy \iint_{w_v} \left(p_{w_v,c_m}\,(x,y) - \overline{p_{w_v,c_m}}\right)^2 dxdy}}, \tag{2.3}$$

where w_v denotes the template window around v (Figure 2.8), $p_{w_v,c}$ the texture corresponding to w_v on the image captured by c, and $\overline{p_{w_v,c}}$ the average of the $p_{w_v,c}$. Note that the template window w_v for v is a rectangle plane tangent to v with a certain size, and is projected onto each camera c to obtain the texture $p_{w_v,c}$ (Figure 2.8). The size of the template window in practice is determined by the distances between neighboring vertices and the resolution of captured images. In this definition, we assume that:

- C_v, the set of visible cameras of v, is constant when v moves slightly to compute, and $\nabla\left(E_e\left(\boldsymbol{q}_v\right)\right)$,
- the small region of the object surface around v can be approximated as plane tangent to v

for the sake of computing performance, and

- object surface is Lambertian,

because it is hard to estimate both the shape and reflectance property simultaneously.

This photometric force $\boldsymbol{F}_e(v)$ moves each vertex v so that

- its corresponding image textures observed by the cameras in C_v become mutually consistent if multiple cameras can observe v, or
- leave v to be moved by other forces if no or only one camera can observe it.

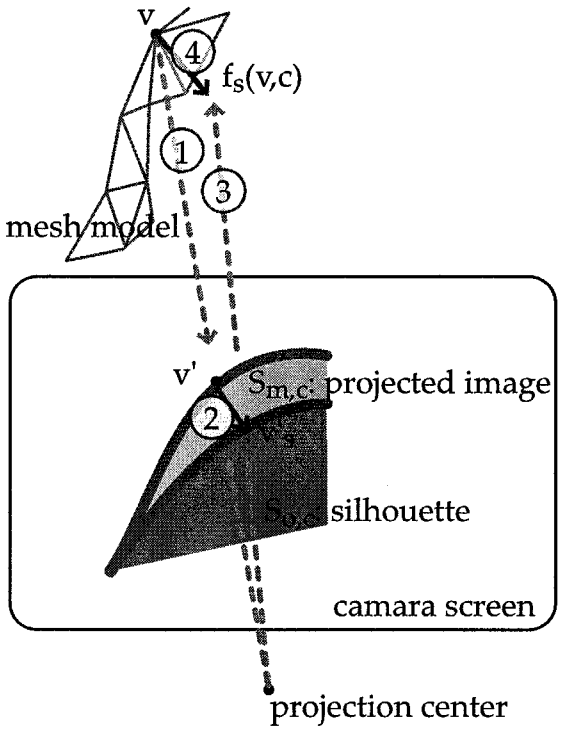


Figure 2.9: Silhouette preserving force

Silhouette Preserving Force

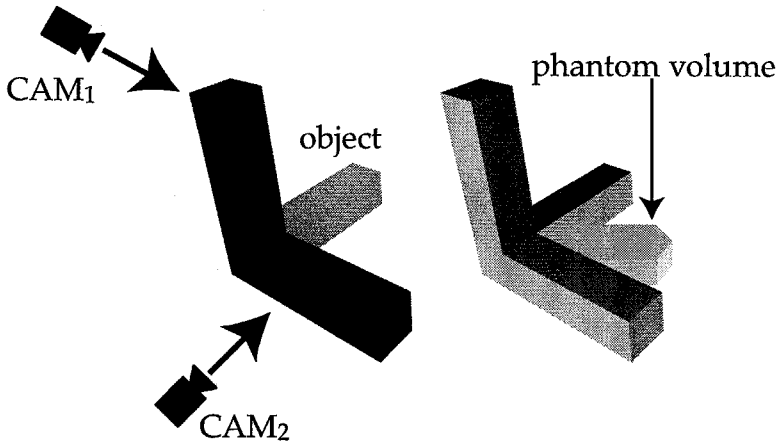
To satisfy the silhouette constraint described above, we introduce a silhouette preserving force $F_s(v)$. This is the most distinguishing characteristic of our deformable model and involves a nonlinear selection operation based on the global shape of the mesh, which cannot be analytically represented by an energy function.

Figure 2.9 explains how this force at v is computed, where $S_{o,c}$ denotes the object silhouette observed by camera c, $S_{m,c}$ the 2D projection of the 3D mesh onto the image plane of camera c, and v' the projection of v onto the image plane of camera c.

1. For each c in C_v, compute the partial silhouette preserving force $f_s(v,c)$ by the following method.

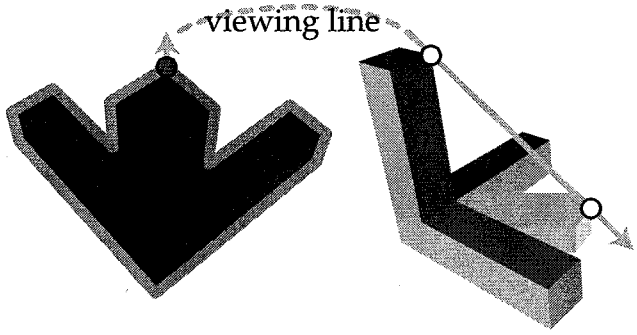
2. If v' is on the boundary of mesh silhouette $S_{m,c}$ between the background, and

   (a) v' is located outside of $S_{o,c}$ or



(a) object and cameras (b) visual hull

(c) captured image at CAM1 (d) captured image at CAM2



(e) silhouette outline and correspondings

Figure 2.10: Multiple contour generator candidates for single apparent contour

(b) v' is located inside of $S_{o,c}$,

then compute the shortest vector from v' to $S_{o,c}$ (Figure 2.9 ②), i.e., v'_s and assign its corresponding vector to $f_s(v,c)$ (Figure 2.9 ④, see below).

3. Otherwise, $f_s(v,c) = 0$.

Here, we compute $f_s(v,c)$ (Figure 2.9 ④) as follows:

$$f_s(v,c) = \left(n_v \cdot d_{v,v'}\right) n_v, \tag{2.4}$$

where n_v denote the normal vector at v and $d_{v,v'}$ the vector from v' to v'_s. Note here that $d_{v,v'}$ is a 3D vector represented in the coordinate system of the mesh model.

The overall silhouette preserving force at v is computed by summing up $f_s(v,c)$:

$$F_s(v) \equiv \sum_{c \in C_v} f_s(v,c). \tag{2.5}$$

Note that $F_s(v)$ acts only on those vertices that are located around the contour generator[CZ00] of the mesh, which is defined based on the global 3D shape of the object as well as the locations of cameras' image planes.

Here, we may have multiple vertices for single point of 2D mesh silhouette $S_{m,c}$ even though we assumed that the cameras are in general position. Suppose we are considering an object illustrated in Figure 2.10(a). From two cameras CAM_1 and CAM_2, the object are observed as illustrated in Figure 2.10(c) and Figure 2.10(d) respectively. The outlines of these two images generate the visual hull of the object with *phantom* volume (in orange) as shown in Figure 2.10(b). Here, phantom means false-positive portions of a visual hull, and it is generated as if shadows of true-positive volumes. Since we use the visual hull as the initial shape of deformation (described later), let us consider the silhouette preserving force for this visual hull. As illustrated in Figure 2.10(e), we have two contour generator candidates (white circles on right) for single point (grey circle on left) of silhouette boundary (bold silver line on left). In this case, partial silhouette preserving force $f_s(v)$ takes both candidates as contour generator, i.e., does not select itself.

However, it is obviously required to do the selection of contour generator to obtain the real object shape. We implement selective operation as local support of contour generator candidates. As we described in Section 2.3.2, contour gener-

ators should be placed continuously on the object surface, and we assumed that only one contour generator should correspond to one point of silhouette outline, we define the local support for a vertex as follows.

Step 1. Initialize the likelihood of each vertex to 0.

Step 2. For each point of silhouette outlines, let each vertex accumulate its likelihood of contour generator. We add the likelihood by the following rule:

- if a silhouette point has n corresponding vertices, add $\frac{1}{n}$ to each vertex likelihood.

For example, if a vertex has 3 another competitors, it gets $\frac{1}{4}$. We denote the competitors of v by $\mathrm{Comp}(v)$.

Step 3. For each vertex, sum up the likelihood values within its neighbors. We denote accumulated value by $S_L(v)$.

Step 4. For each vertex, compute the modified local support $L(v)$:

$$L(v) = \frac{S_L(v)}{S_L(v) + \sum_{v' \in \mathrm{Comp}(v)} S_L(v')}. \tag{2.6}$$

This function selects a vertex which has higher local support from its competitors.

With this local support $L(v)$, we modify $\boldsymbol{F}_s(v)$ as follows:

$$\boldsymbol{F}_s(v) \equiv L(v) \sum_{c \in \mathrm{C}_v} f_s(v, c). \tag{2.7}$$

Internal Force

Since $\boldsymbol{F}_e(v)$ may destroy the smoothness of the mesh or lead to self-intersection, we introduce an internal force $\boldsymbol{F}_i(v)$ at v:

$$\boldsymbol{F}_i(v) \equiv \frac{\sum_j^n \boldsymbol{q}_{v_j} - \boldsymbol{q}_v}{n}, \tag{2.8}$$

where $\boldsymbol{q}_{v_j}$ denotes the neighboring vertices of v and n the number of neighbors. $\boldsymbol{F}_i(v)$ act as tension between vertices and keeps them locally smooth.

Note that the utilities of this internal force is twofold:

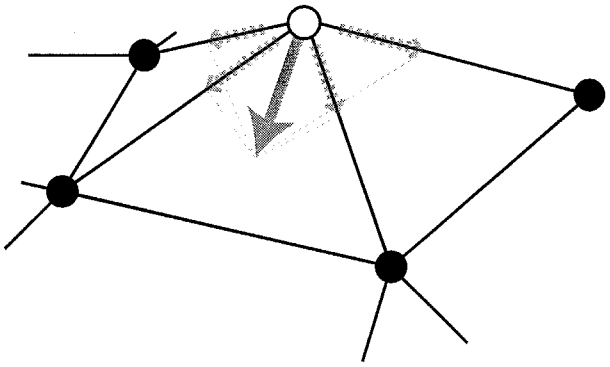


Figure 2.11: Internal force

1. make the mesh shrink, and

2. make the mesh smooth.

We need 1. in the intra-frame deformation since it starts with the visual hull which encases the real object shape (described below). 2. on the other hand, stands for a popular smoothness heuristic employed in many vision algorithms such as the regularization and active contour models. The smoothing force works to prevent self-intersection since a self-intersecting surface includes protrusions and dents, which will be smoothed out before causing self-intersection.

For the inter-frame deformation, on the other hand, we redefine $\boldsymbol{F}_i(v)$ as a combination of attraction and repulsion between linked vertices (see Chapter 3). This is because we do not need (1) described above in the inter-frame deformation process.

Overall Vertex Force

Finally we define a vertex force $\boldsymbol{F}(v)$ with coefficients α, β, γ as follows:

$$\boldsymbol{F}(v) \equiv \alpha \boldsymbol{F}_i(v) + \beta \boldsymbol{F}_e(v) + \gamma \boldsymbol{F}_s(v), \tag{2.9}$$

where coefficients are constant and examples for typical values will be given in following experiments section. $\boldsymbol{F}_e(v)$ and $\boldsymbol{F}_s(v)$ work to estimate the accurate object shape and $\boldsymbol{F}_i(v)$ to smooth and interpolate the shape. Note that there may be some vertices where $C_v = \emptyset$ and hence $\boldsymbol{F}_e(v) = \boldsymbol{F}_s(v) = 0$.

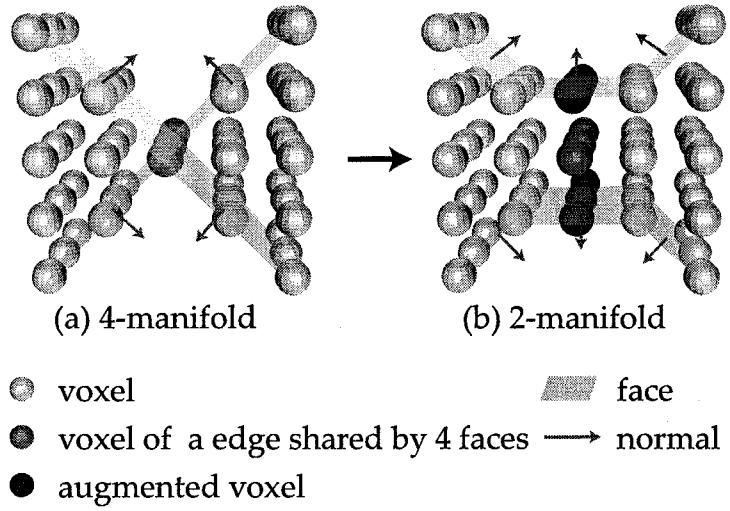


Figure 2.12: Conversion of four-manifold surface to two-manifold

2.4.3 Computation Scheme

We defined the forces to move vertices so as to satisfy the constraints which should be satisfied if the mesh model represents the real object shape. So if we can obtain a mesh model such that the forces on each vertex are balanced, we assume that the balanced mesh model is the estimation result of the object shape.

Initial shape

As described in Section 1.2, the visual hull encages the object by its definition. So if we assume that we can obtain silhouettes on each viewpoint accurately enough, it is reasonable to use the visual hull as the initial shape of the deformable mesh model.

In general, since we can obtain the visual hull as a set of voxels[MWTN04], we need to convert it into surface representation. We employed discrete marching cubes method[KKI99]. This is because:

- It is mathematically proven to generate a closed, two- or four-manifold discrete surface, and
- It generates unique triangle mesh.

Note that four-manifold is not a problem. If we detect such points, we only have to change the sampling rate of the voxel space to higher, and then re-convert into surface representation. This is because that four-manifold is generated if the

sampling rate of voxel space is too sparse. Also, in practice, we can expand such points in voxel space and re-convert it again. That is,

Step 1. Suppose we have a voxel space V. We denote each voxel at (x,y,z) by $V(x,y,z)$, and assign $V(x,y,z) = 1$ if the voxel is occupied, and $V(x,y,z) = 0$ if empty. The object shape is represented as the set of voxels such that its value is 1.

Step 2. Convert V into surface representation M by [KKI99].

Step 3. For each edge in M, check if it is shared by four triangles or not. Let E_4 denote the set of edges each of which are shared by four triangles.

Step 4. If $E_4 \neq \emptyset$, do the following and then go back to Step 1.

- For each vertex of edges in E_4, let us denote its position by (x,y,z) (green spheres in Figure 2.12).
- Then, for each voxel in the 18-neighborhood of a voxel at (x,y,z), set its value to 1 (red spheres in Figure 2.12).

Step 5. Now, we have a mesh model such that $E_4 = \emptyset$, i.e., a two-manifold mesh model.

Iterative computation

To find the optimal, force-balanced state of the mesh, we use greedy iterative method:

Step 1. Compute the force $\boldsymbol{F}_v$ acting on each vertex.

Step 2. Move each vertex according to the force.

Step 3. Terminate if all vertex motions are small enough. Otherwise go back to Step 1.

This is because

1. we have too many vertices to solve directly, and
2. the forces on each vertex affects each other; if a vertex moves, visibilities (C_v) of others may change. So we cannot subdivide this optimization problem into sub-problems.

Note that in each iteration, we assume that visibilities of each vertex are fixed and we can compute the forces independently by limiting the vertex motion to be short enough.

This iterative computation is similar to a simulation of physics that computes temporal deformation of an object shape under forces working on object surfaces.

2.5 Overall Algorithm

Overall algorithm for the intra-frame deformation is as follows:

Input Multi-viewpoint object images and silhouettes.

Output 3D shape of the object represented by triangle patches.

Step 1. Capture the object and obtain the silhouettes at each viewpoint.

Step 2. Acquire the visual hull of the object as a set of voxels from its multi-viewpoint silhouettes by visual cone intersection method.

Step 3. Obtain the initial shape of the mesh model by converting the visual hull into triangle patches with discrete marching cubes method.

Step 4. Deform the mesh iteratively:

- Step 4.1. Compute the visibilities of each vertex.
- Step 4.2. Suppose the visibilities are fixed, and compute the forces F_v working on each vertex independently.
- Step 4.3. Move each vertices according to F_v.
- Step 4.4. Terminate if all vertex motions are small enough. Otherwise go back to Step 4.1.

Step 5. Take the deformed mesh as the estimation result.

2.6 Performance Evaluation

2.6.1 Reconstruction from Synthesized Images

To evaluate the quantitative performance of the mesh model, we conducted experiments with synthetic objects defined by super quadric functions. Super

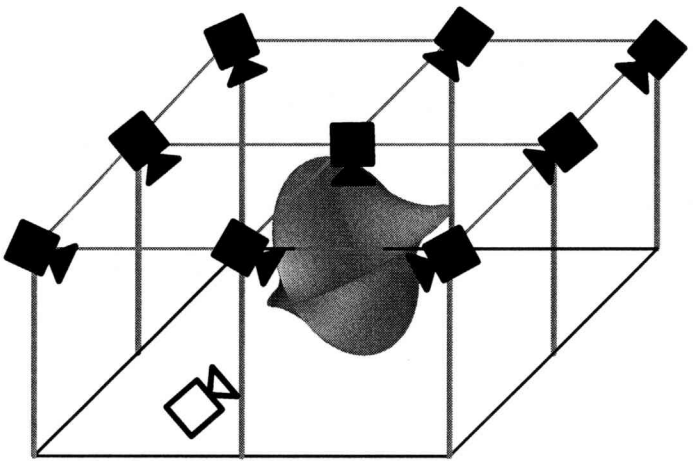
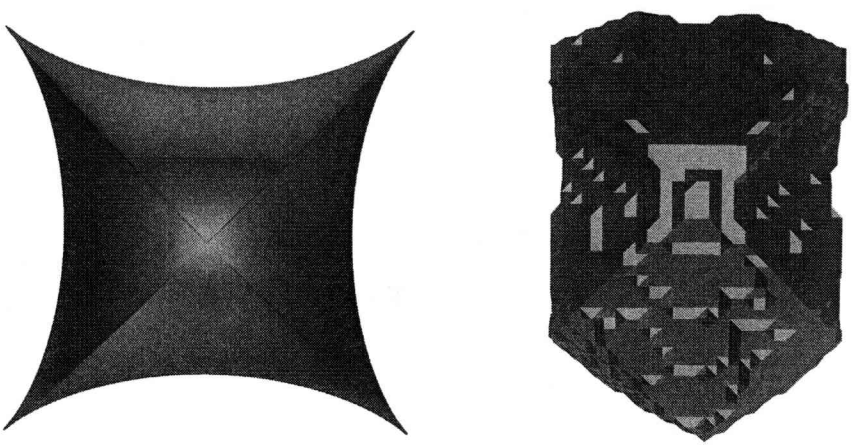
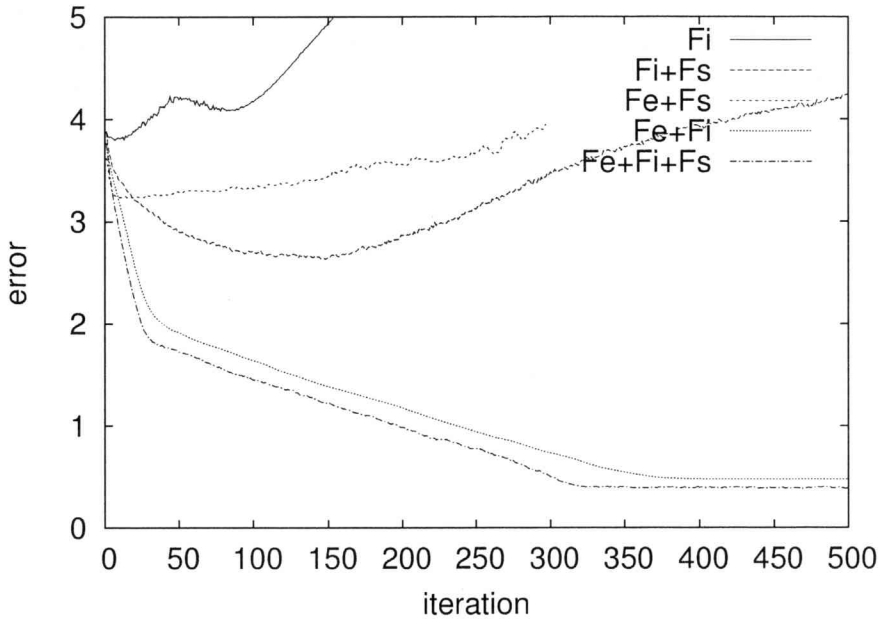


Figure 2.13: Camera arrangement

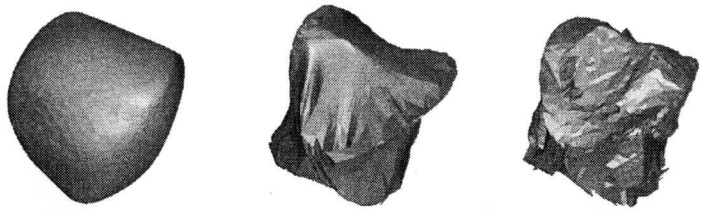


(a) Synthesized object (b) Visual hull

Figure 2.14: Synthesized object and visual hull



(a) error



(b)$\boldsymbol{F}_i$ (c)$\boldsymbol{F}_i + \boldsymbol{F}_s$ (d)$\boldsymbol{F}_e + \boldsymbol{F}_s$

(e)$\boldsymbol{F}_e + \boldsymbol{F}_i$ (f)$\boldsymbol{F}_e + \boldsymbol{F}_i + \boldsymbol{F}_s$

Figure 2.15: Evaluating the effectiveness of each force

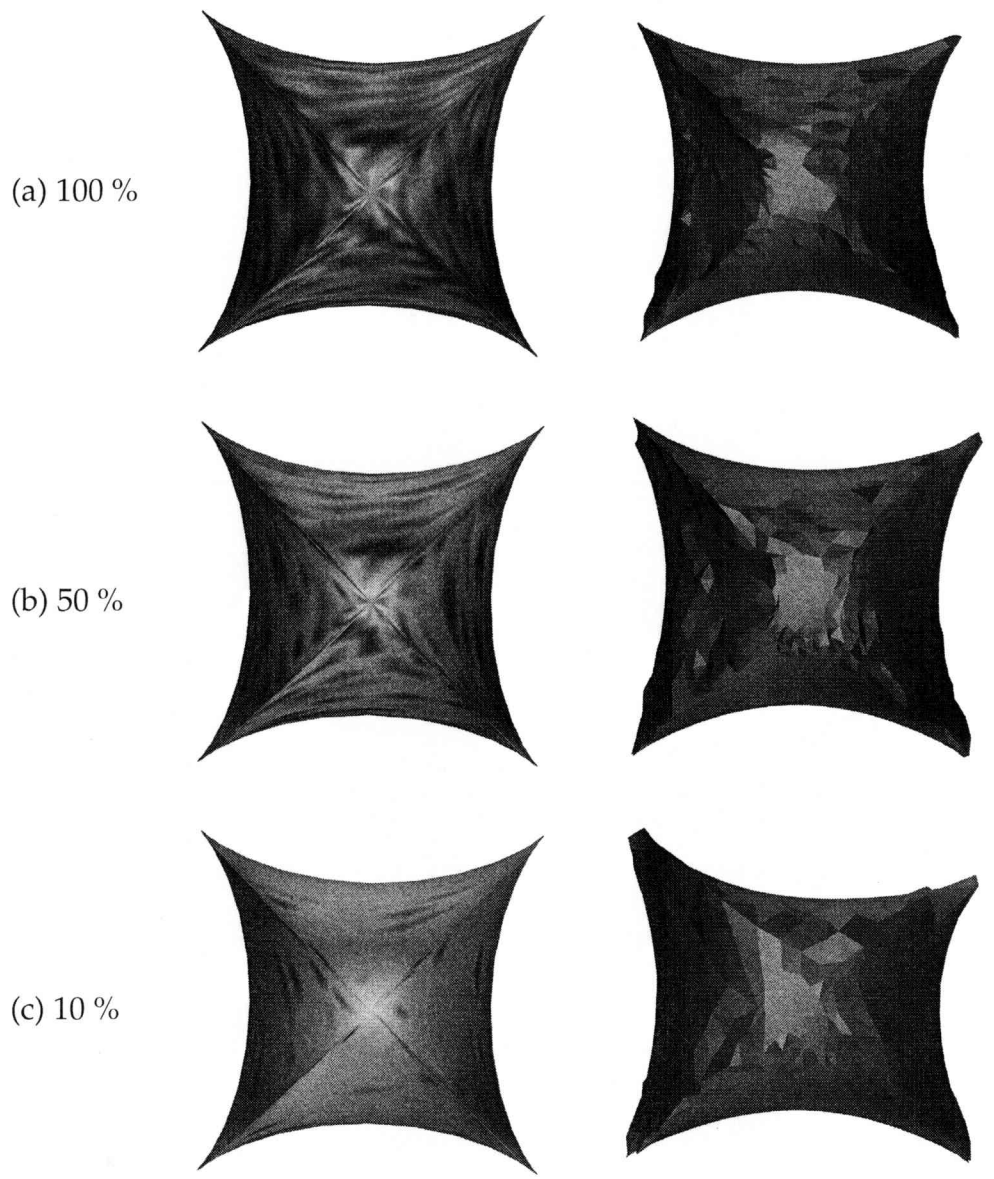


Figure 2.16: Evaluating the effectiveness of the deformable mesh model.
Left: super quadrics ($n = 3.0$, $e = 1.0$) with different degrees of surface textures: k % means k % of the surface area is covered with texture. Right: reconstructed object shapes.

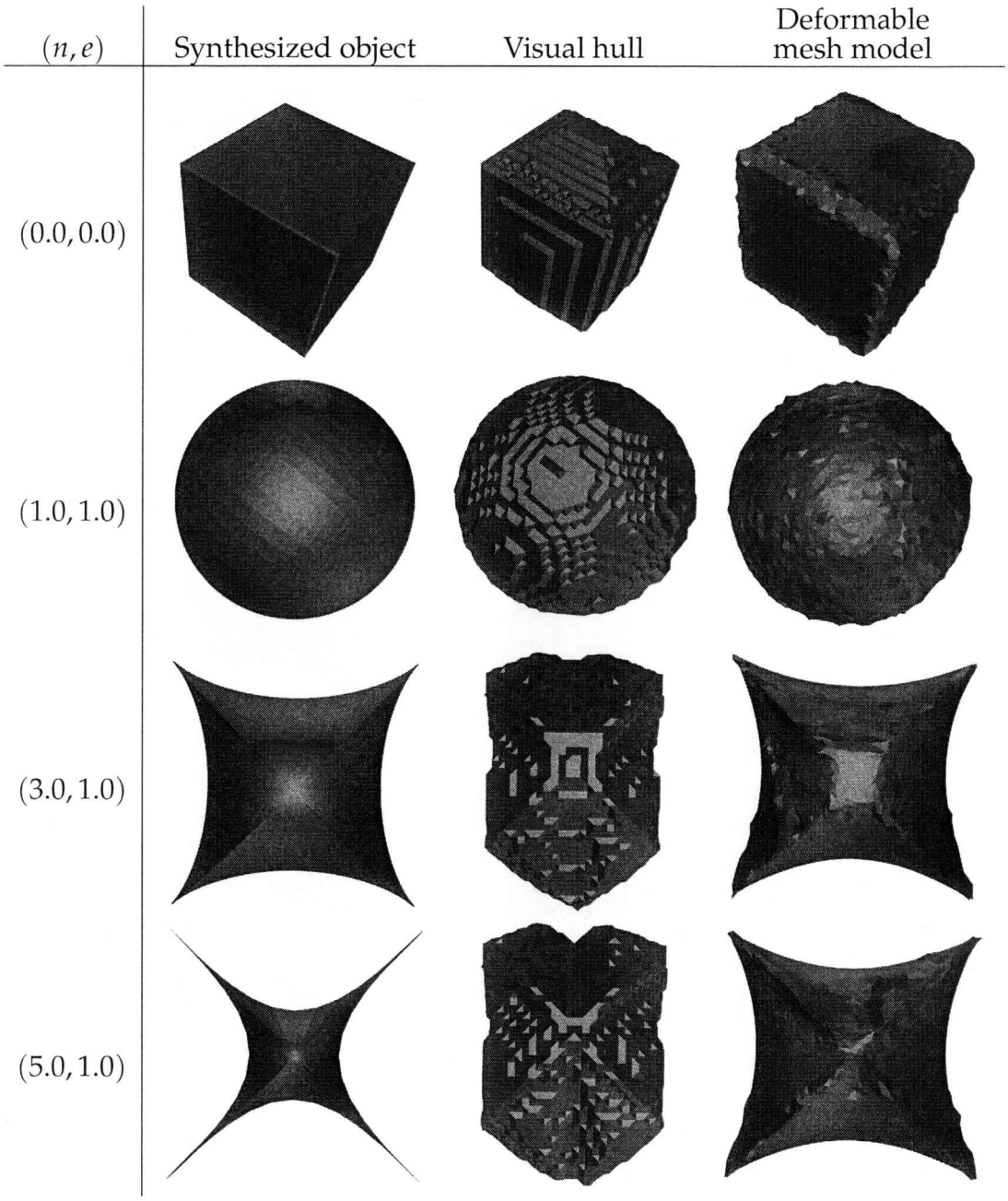


Figure 2.17: Reconstruction results for various n and e (with 100% textured surfaces)

quadric functions are a set of functions defined in terms of spherical coordinates u and v:

$$\begin{aligned} x(u,v) &= a_1 \cos^n u \cos^e v \\ y(u,v) &= a_2 \cos^n u \sin^e v \\ z(u,v) &= a_3 \sin^n u \\ -\frac{\pi}{2} \leq u \leq \frac{\pi}{2},\ & -\pi \leq v \leq \pi \end{aligned} \tag{2.10}$$

where n and e denote parameters controlling roundness/squareness, a_1, a_2, a_3 denote scale factors for x, y, z respectively. In this experiment, the values of the coefficients α, β, and γ are the same as used in the experiment with real images, that is, $\boldsymbol{F}(v) = 0.3\boldsymbol{F}_i(v) + 0.4\boldsymbol{F}_e(v) + 0.3\boldsymbol{F}_s(v)$.

Figure 2.13 illustrates the camera arrangement for this experiment, Figure 2.14 (a) the synthesized object, and (b) the visual hull reconstructed by the volume intersection method. We use the 9 black cameras in Figure 2.13 for the reconstruction and the white camera in Figure 2.13 for the evaluation.

Figure 2.15 shows reconstruction results of objects (Figure 2.14 (a)) with the following configurations: (b)$\boldsymbol{F}(v) = 0.3\boldsymbol{F}_i$, (c)$\boldsymbol{F}(v) = 0.3\boldsymbol{F}_i + 0.3\boldsymbol{F}_s$, (d)$\boldsymbol{F}(v) = 0.4\boldsymbol{F}_e + 0.3\boldsymbol{F}_s$, (e)$\boldsymbol{F}(v) = 0.4\boldsymbol{F}_e + 0.3\boldsymbol{F}_i$, and (f)$\boldsymbol{F}(v) = 0.4\boldsymbol{F}_e + 0.3\boldsymbol{F}_i + 0.3\boldsymbol{F}_s$. We used the visual hull of the object (Figure 2.14(b)) as the initial mesh model. The graph in Figure 2.15(a) shows how the average error between the mesh model and the real shape changes during the iterative shape deformation. Note that the mesh models are rendered in grey, but they have randomly generated textures in the experiments. This is because that if it is rendered with textures, it becomes hard to observe shape errors. We can observe that

- Without the internal force $\boldsymbol{F}_i$, a lot of self-intersections breaks down the deformation since we cannot estimate the visible cameras for each vertex appropriately.
- The photometric force $\boldsymbol{F}_e$ can estimate concave positions which cannot be estimated by the visual hull.
- The silhouette force $\boldsymbol{F}_s$ can preserve contour generators i.e., the outline of the object (Figure 2.15(e) and (f)).

Figure 2.16 shows reconstruction results of objects ($n = 3.0$, $e = 1.0$) with (a) 100%, (b) 50%, and (c) 10% textured surfaces. The percentage denotes the surface

area of the object covered with texture. All images of the synthesized object in figures are captured by the white camera in Figure 2.13.

From these results we can observe the following:

- Unlike the visual hull (Figure 2.14(b)), the mesh model can reconstruct the concave parts of the object (Figure 2.16(a) and (b)).
- The mesh model does not necessarily require a dense texture (Figure 2.16(a) and (b)). This is because the *skin* over *frames* (Figure 2.7) can interpolate the object surface between points with prominent textures.
- The reconstructed shape becomes poor when the object has little texture (Figure 2.16(c)).

Figure 2.17 shows reconstruction results of objects defined by various n and e, that is, objects having different concavities. Note that each object has 100% textured surface.

We can observe that the deformable mesh model with fixed coefficients α, β, and γ has limitations in recovering large concavities as well as large protrusions (Figure 2.17, bottom row). This is because large curvature on a vertex yields a large internal force $\boldsymbol{F}_i(v)$, which dominates $\boldsymbol{F}_e(v)$ even if the vertex has prominent texture.

2.6.2 Reconstruction from Real Images

Figure 2.18 illustrates the camera arrangement for the experiments, where we use $\mathrm{CAM}_1, \ldots, \mathrm{CAM}_4$ for shape reconstruction and CAM_5 for performance evaluation. That is, we compare the 2D silhouette of the reconstructed shape viewed from the position of CAM_5 with the actually observed by CAM_5. Note that captured images are synchronized and blur-free.

Figure 2.20 shows the initial object shape computed by the volume intersection using the images captured by $\mathrm{CAM}_1, \ldots, \mathrm{CAM}_4$, i.e., the visual hull of the object. The shaded regions of (a) and (b) show the projection of the initial shape, that is, $S_{m,5}$ and $S_{m,1}$, respectively. Bold lines in the figures highlight the contours of $S_{o,5}$ and $S_{o,1}$. We can observe some differences between $S_{o,5}$ and $S_{m,5}$, but not between $S_{o,1}$ and $S_{m,1}$. This is because the image captured by CAM_5 is not used for the reconstruction.

In the experiments, we evaluated our algorithm with the following configurations : (a) $\boldsymbol{F}(v) = \boldsymbol{F}_i(v)$, (b) $\boldsymbol{F}(v) = 0.5\boldsymbol{F}_i(v) + 0.5\boldsymbol{F}_s(v)$, (c) $\boldsymbol{F}(v) = 0.3\boldsymbol{F}_i(v) +$

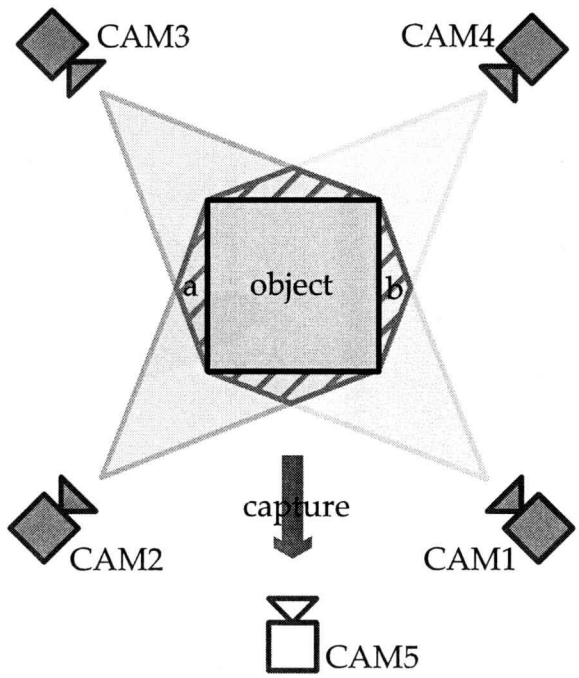
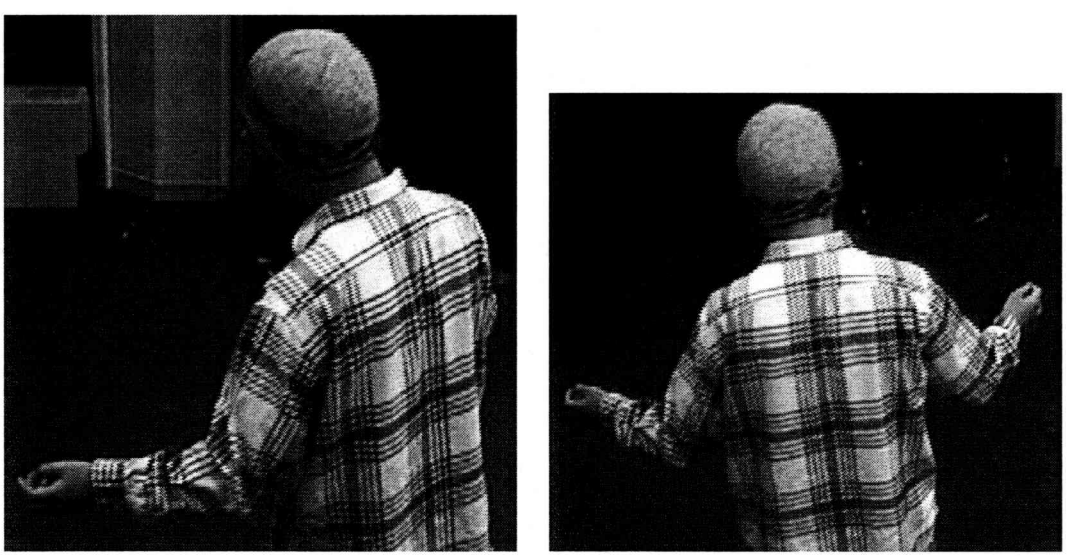
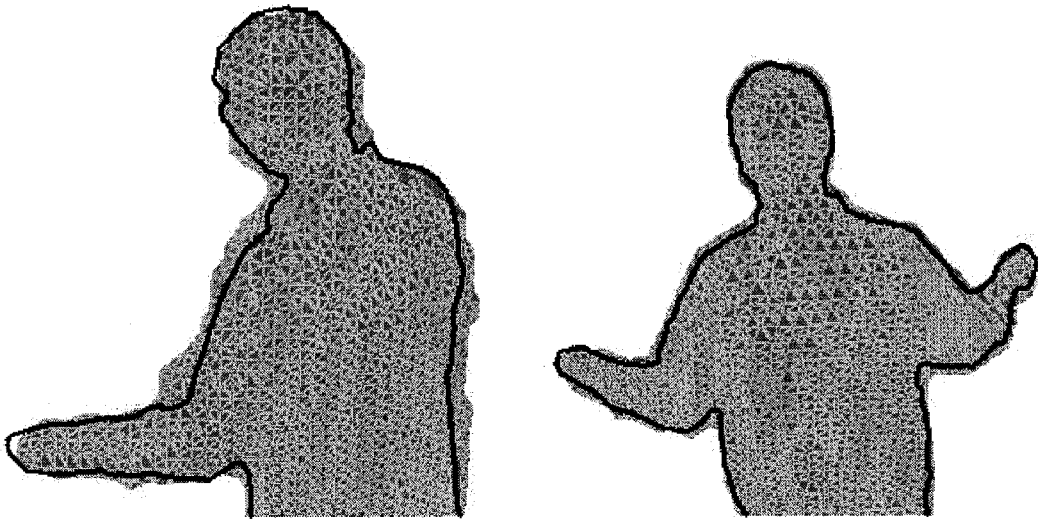


Figure 2.18: Camera arrangement



(a) CAM_5 (b) CAM_1

Figure 2.19: Input images.
(a) captured from CAM_5 in Figure 2.18, (b) from CAM_1



(a) CAM_5 (b) CAM_1

Figure 2.20: Initial shape of the object (visual hull).
(a) captured from CAM_5 in Figure 2.18, (b) from CAM_1

$0.4\boldsymbol{F}_e(v) + 0.3\boldsymbol{F}_s(v)$. The left and center columns of Figure 2.21 shows $S_{m,5}$ and $S_{m,1}$ for each configuration together with bold lines denoting the corresponding observed object silhouette contours $S_{o,5}$ and $S_{o,1}$. The graphs in the right column show how the average error between $S_{m,c}$ and $S_{o,c}$ $(c = 1, 5)$ changes during the iterative shape deformation. Note that the processing time of deformation is about 3 minutes for 12000 vertices and 4 cameras.

From these results we can make the following observations:

- With $\boldsymbol{F}_i(v)$ alone (Figure 2.21(a)), the mesh model shrinks, resulting in a large gap between its 2D silhouette on each image plane and the observed silhouette.
- With $\boldsymbol{F}_i(v)$ and $\boldsymbol{F}_s(v)$, while $S_{m,c}, c = \{1 \ldots 4\}$ match well with $S_{o,c}$, $S_{m,5}$, whose corresponding image is not used for the reconstruction, does not deform well (Figure 2.21(b)).
- With $\boldsymbol{F}_i(v)$, $\boldsymbol{F}_e(v)$, and $\boldsymbol{F}_s(v)$, $S_{m,5}$ matches well with $S_{o,5}$ (Figure 2.21(c)). This shows the effectiveness of $\boldsymbol{F}_e(v)$.

Note that the values of the coefficients α, β, and γ are given a priori and fixed throughout the iteration.

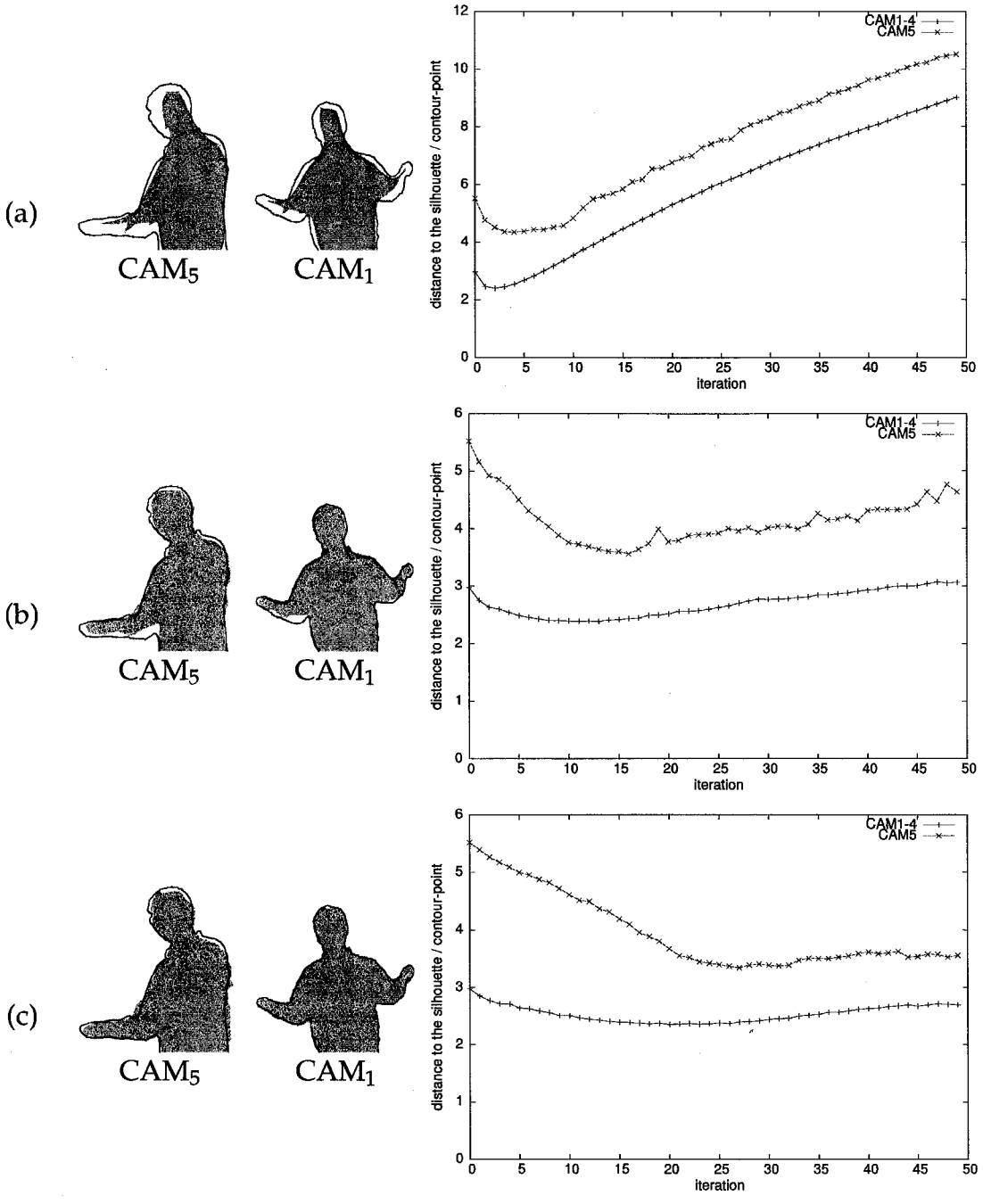


Figure 2.21: Reconstruction results.
Top: (a) $F_i(v)$ alone ($\alpha = 1.0, \beta = 0.0, \gamma = 0.0$), Middle: (b) $F_i(v) + F_s(v)$ ($\alpha = 0.5, \beta = 0.0, \gamma = 0.5$), Bottom: (c) $F_i(v) + F_e(v) + F_s(v)$ ($\alpha = 0.3, \beta = 0.4, \gamma = 0.3$)

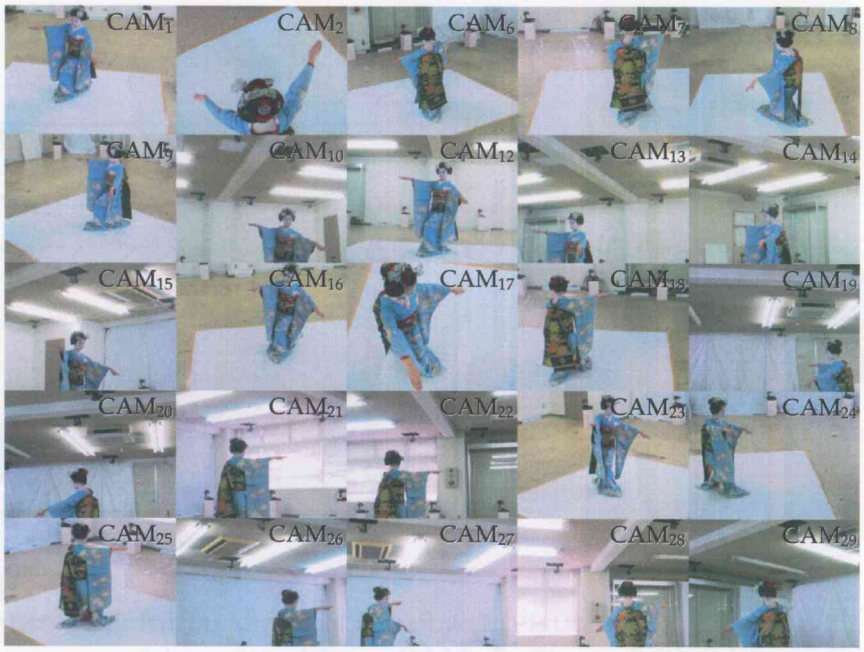


Figure 2.22: Input images

2.6.3 Reconstruction of Complicated Object from Real Images

This experiment shows an example of shape reconstruction of complicated real shape. Figure 2.22 shows input images taken by 25 cameras arranged as illustrated in Figure 2.23. Figure 2.24 shows the visual hull of the object, and we use this as the initial shape of the deformation. This visual hull has about 30,000 vertices and 60,000 triangles, and the length of edges are almost 1cm. Note that this visual hull has a huge false-positive volume in front of its body. This is because its body, sleeves, and skirt form a *cup* shape, and no cameras can observe it as silhouette contour.

Figure 2.25 illustrates the initial shape of the mesh model and the deformation result: (a) the initial mesh model, (b) its rendering result, (c) the deformation result, and (d) its rendering result. Note that we used following algorithm for rendering:

1. For each vertex v, select the most facing camera c_v from visible cameras C_v.
2. Project each v onto the screen of c_v and pick the color at projected point as the color of v.
3. For each triangle patch, fill its internal color by interpolating from its three vertices.

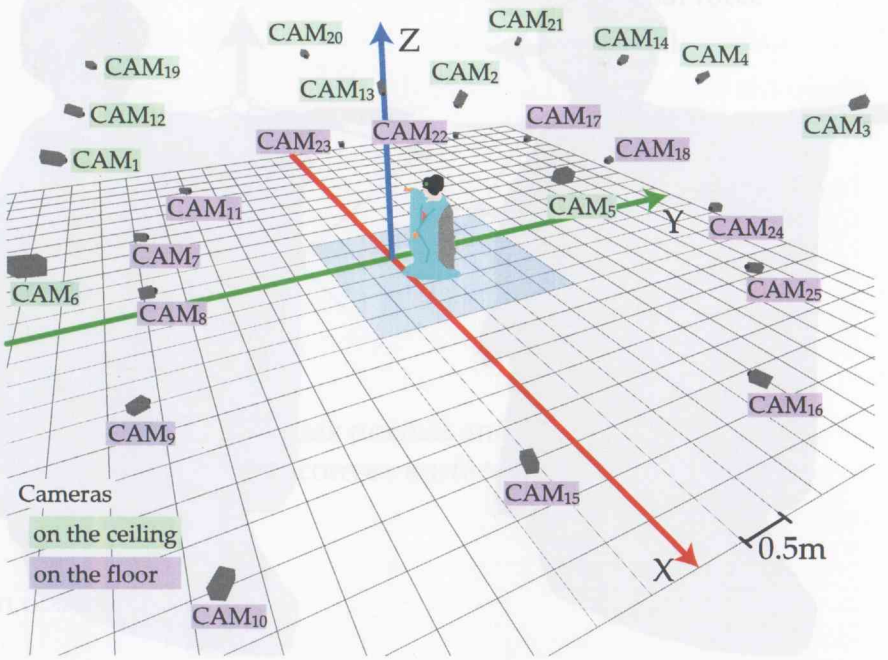


Figure 2.23: Camera arrangement

Figure 2.24: Visual hull

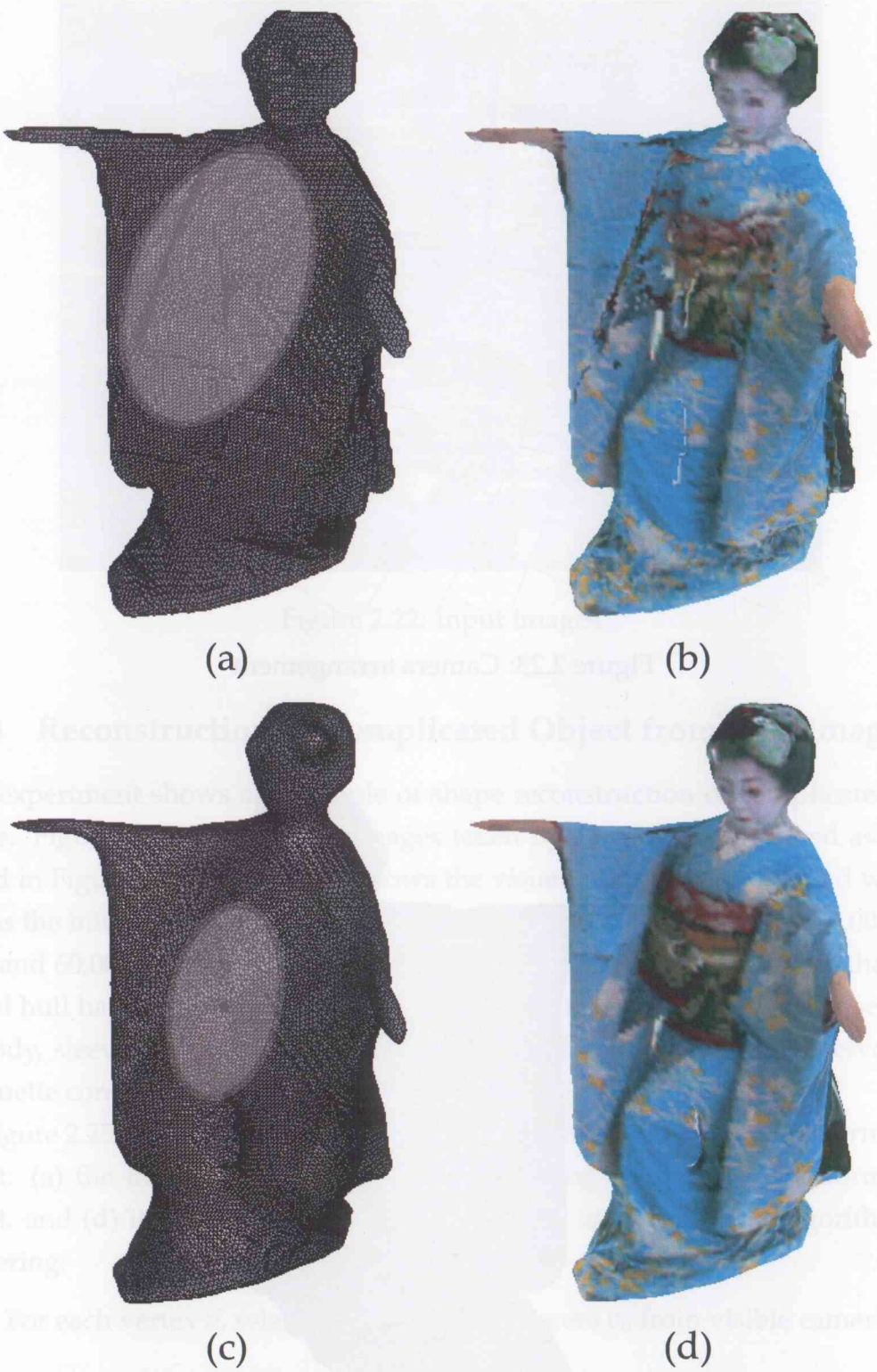


Figure 2.25: Rendering result.
(a) and (b): the initial mesh and its rendering result,
(c) and (d): deformed mesh and its rendering result.

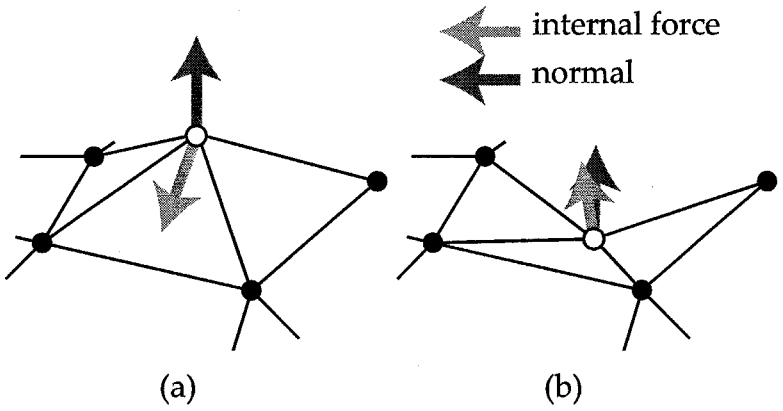


Figure 2.26: Vertex normal and internal force.
(a) convex surface, (b) concave surface

We can observe that:

- The outlines of the mesh model is preserved after the deformation.
- The concave position in front of the object is not reconstructed well. In Figure 2.25(a) and 2.25(c), shadowed ellipses indicate false-positive concave portions, and In Figure 2.25(b) and 2.25(d) we can observe that corresponding regions are rendered poorly while Figure 2.25(d) looks better than Figure 2.25(b). This is because the fixed coefficients of internal and photometric force prevents the mesh surface to exceed a certain concavity as discussed in Section 2.6.1 with synthesized object.

To overcome this limitation, we next introduce an adaptive control algorithm of the force coefficients.

2.7 Adaptive Control of the Force Coefficient for Concavity

As is well known, it have been a common topic in active contour models how it cope with a concavity of the object[XP98]. That is, naive implementation of active contour model has a limitation about the shape concavity since its smoothness constraint forces the model to be flat basically. To overcome this problem, there are mainly two points to discuss – one is how and the other is when we should make the mesh model be able to exceed such a limited concavity.

There are two approaches proposed to allow an over-concavity. One add a node and the other reduce the ratio of an internal force. In this thesis, we employ the latter approach because we consider that it is very valuable to preserve the mesh topology. In the following chapters, we introduce an algorithm to estimate the object shape and motion in two frames, and its main benefit is to produce fully per-vertex correspondences between two frames. This full correspondence is strongly required to compress the meshes by geometry videos[BSM+03][IR03] which is a typical application of our algorithm.

The next topic is when we should reduce the coefficient of internal force β. First, we can detect if the vertex v is located at concave part of the mesh surface with its normal vector $\boldsymbol{n}_v$. We defined the internal force $\boldsymbol{F}_i(v)$ as the vector toward the gravity of its local neighbors:

$$\boldsymbol{F}_i(v) \equiv \frac{\sum_j^n \boldsymbol{q}_{v_j} - \boldsymbol{q}_v}{n}, \tag{2.11}$$

where $\boldsymbol{q}_{v_j}$ denotes the neighboring vertices of v and n the number of neighbors (Equation (2.8)). It is intuitively obvious that if the normal of v and the internal force are in the opposite direction, the vertex is at convex mesh surface (Figure 2.26(a)). On the other hand, if the normal of v and the internal force are in the same direction, the vertex is at concave mesh surface (Figure 2.26(b)). With this concavity detection, we simply introduce following criterion:

> When the internal force of a vertex and the surface normal are in same the direction, and the photometric force indicates the opposite direction, and no silhouette force works on it, we reduce the coefficient of the internal force β.

That is,

$$\frac{\boldsymbol{n}_v}{\|\boldsymbol{n}_v\|} \cdot \frac{\boldsymbol{F}_i(v)}{\|\boldsymbol{F}_i(v)\|} < 0 \text{ and } \frac{\boldsymbol{F}_e(v)}{\|\boldsymbol{F}_e(v)\|} \cdot \frac{\boldsymbol{F}_i(v)}{\|\boldsymbol{F}_i(v)\|} < 0 \text{ and } \boldsymbol{F}_s(v) = 0.$$

Note that we should not apply the adaptive control where the silhouette force works to keep the apparent contour.

Using this criterion, we modify the deformation process as follows:

Step 1. Compute the force $\boldsymbol{F}_v$ acting on each vertex.

Step 2. If

- the normal $\boldsymbol{n}_v$ and $\boldsymbol{F}_i(v)$ are in the same direction, and

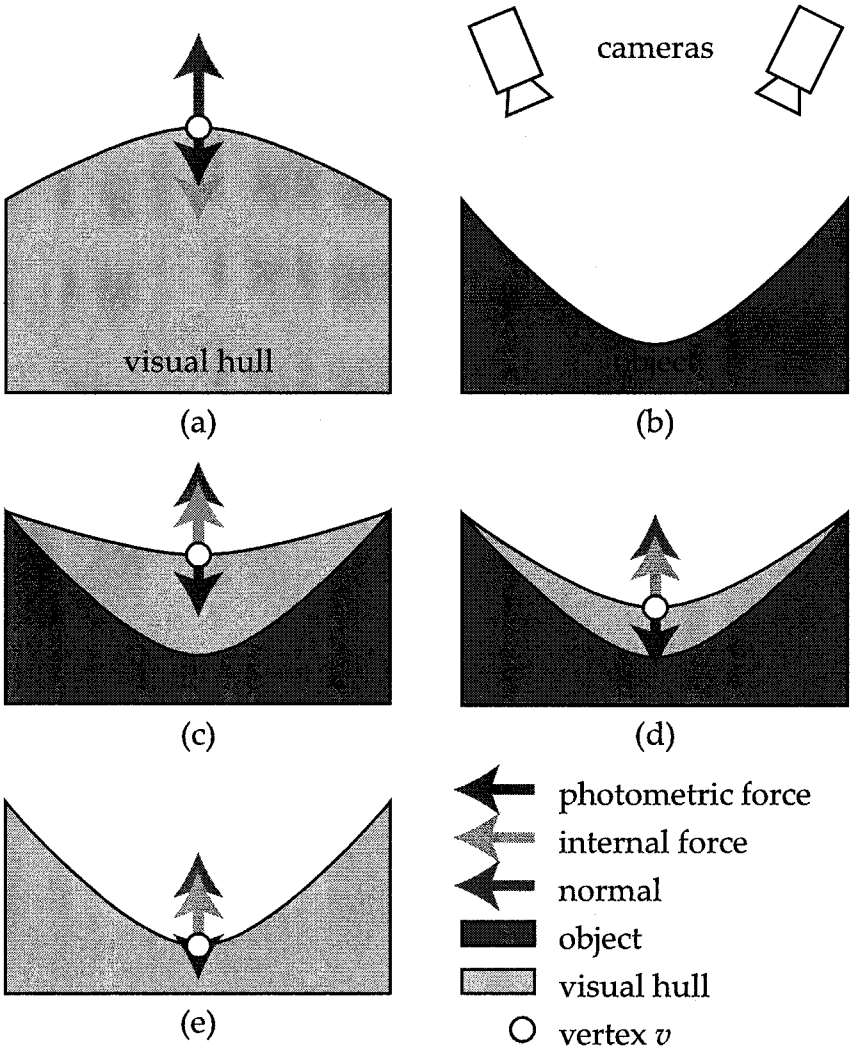


Figure 2.27: Adaptive control of the coefficient

- $F_e(v)$ indicates the opposite direction, and
- $F_s(v) = 0$,

let the coefficient of the internal force β be smaller.

Step 3. Move each vertex according to the force.

Step 4. Terminate if all vertex motions are small enough. Otherwise go back to Step 1.

Figure 2.27 illustrates how this algorithm works.

1. Suppose we start the deformation with the shape in Figure 2.27(a), and deform it so as to be the real object shape in Figure 2.27(b).

2. At the first of the deformation, the surface normal and the internal force of v are in the opposite direction.

3. If we keep the coefficients be constant, the deformation will stop when the photometric force and the internal force balance out (2.27(c)). Here, the normal and the internal force are in the same direction.

4. If the object has prominent textures and the photometric force indicates the opposite direction, that is, more concavity, the coefficient of the internal force β became smaller and let the vertex move according to the photometric force.

5. Smaller β enables the mesh to represent more concave surface, but it will introduce another limitation and the deformation will stop at a certain concavity (2.27(d)).

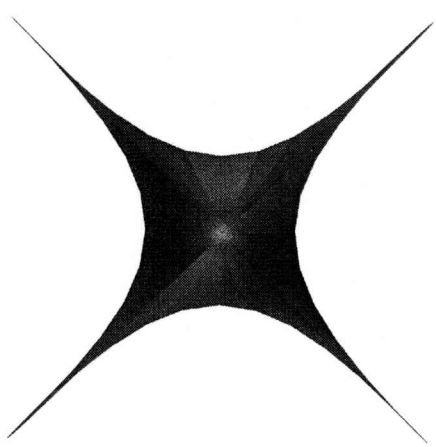
6. However, as far as the photometric force indicates better location, β became smaller and smaller. So if the object surface is well textured, the mesh finally reaches there (2.27(e)).

2.7.1 Performance Evaluation

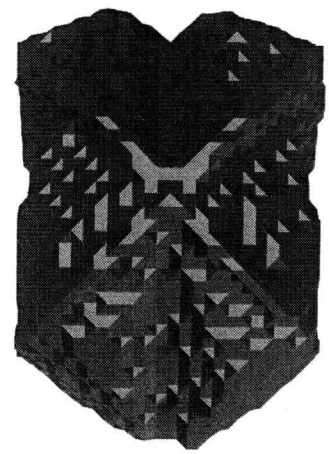
Reconstruction from Synthesized Images using Adaptive Coefficient Control-from

To evaluate the effective of the adaptive control of the coefficient, we show the experimental results of shape reconstruction for synthesized object with concave portions shown in Figure 2.28(a). Note that the camera arrangement and other situations are same as Section 2.6.1.

Compared with the deformation result with fixed coefficient (Figure 2.28(c)), deformation with adaptive coefficient control gives better result as shown in Figure 2.28(d). The graph in Figure 2.29 shows how the average error between the deformable mesh model and the synthesized shape changes during the iterative shape deformation. Note again that as described in Section 2.6.1, the mesh models are rendered in grey, but they have randomly generated textures in the experiments.

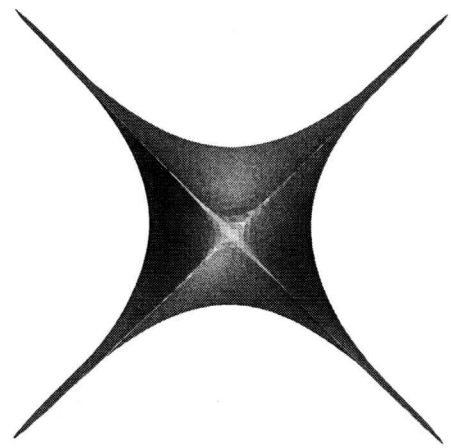


(a) Synthesized object
$(n, e) = (5.0, 1.0)$



(b) Initial Shape
(visual hull)

(c) Deformation result
with fixed coefficient



(d) Deformation result
with adaptive coefficient

Figure 2.28: Deformation result with adaptive coefficient control

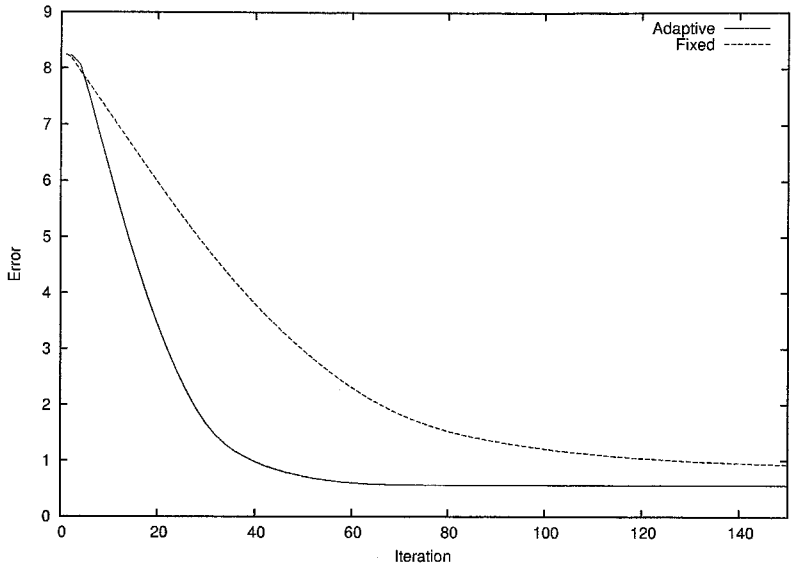


Figure 2.29: Shape error

Reconstruction of Complicated Object with Concavities from Real Images using Adaptive Coefficient Controlfrom

Next, we show the experimental results of shape reconstruction for complicated real shape which was used in Section 2.6.3. The camera arrangement, input images, and the initial shape of the deformation is as same as in Section 2.6.3 (Figure 2.22, 2.23, 2.24).

Figure 2.30 shows deformation results. In Figure 2.30(a), we can observe that concave portion is well estimated than the result of fixed coefficient deformation (Figure 2.25), however, there are still false-positive region indicated by shadowed circle in Figure 2.30(a). We can also observe this false-positive region as poor rendering in Figure 2.30(b). This is because that the lower side of the belt is entirely in black, and the vertices placed nearby there are also in photometrically consistent state.

Figure 2.32 shows rendering result of deformed mesh models. All images in this figure is rendered as if they are observed from CAM_{12} in Figure 2.23 which is closest to the concave region at the front of the object. The top row of Figure 2.32 shows deformation results with fixed (left) and adaptive (right) coefficients. The middle and bottom rows show the generated images rendered by the algorithm described in Section 2.6.3. Note that the images at the bottom row are rendered

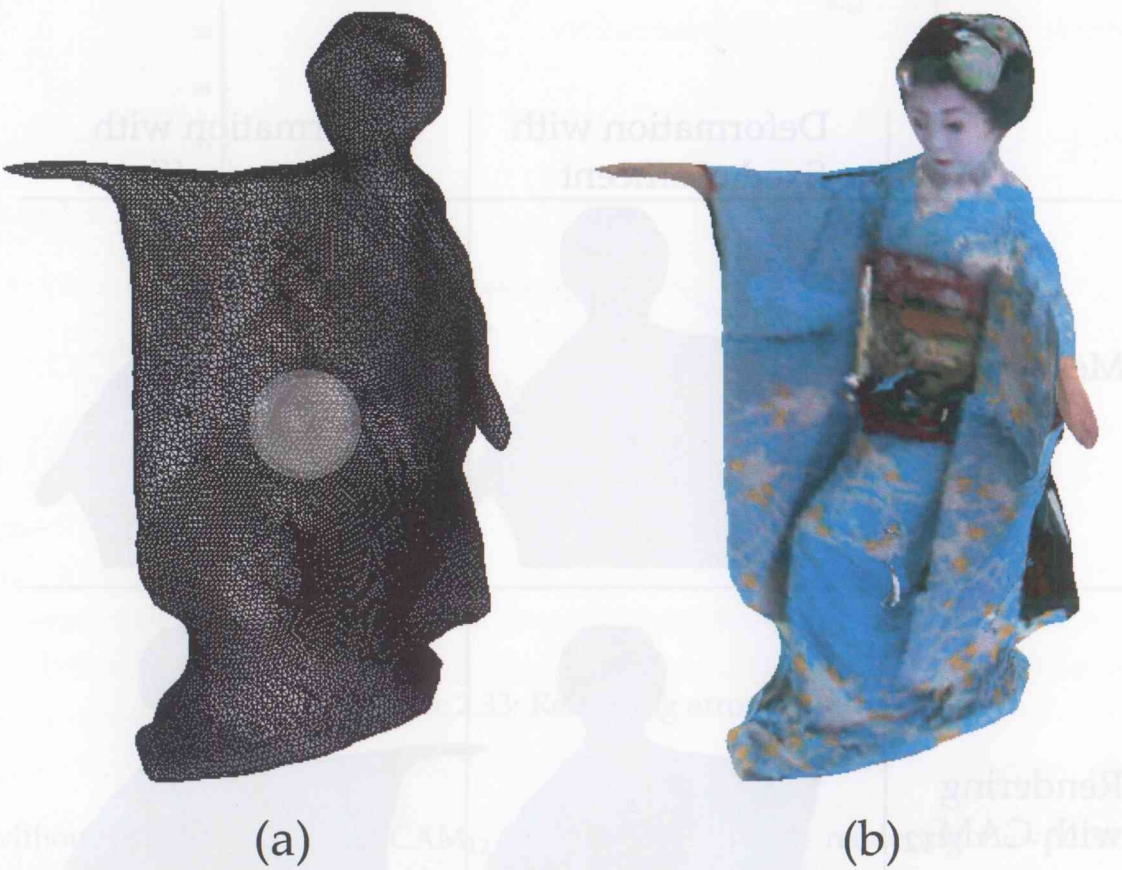


Figure 2.30: Deformed shape with adaptive coefficient control.
(a) mesh model, (b) rendering result

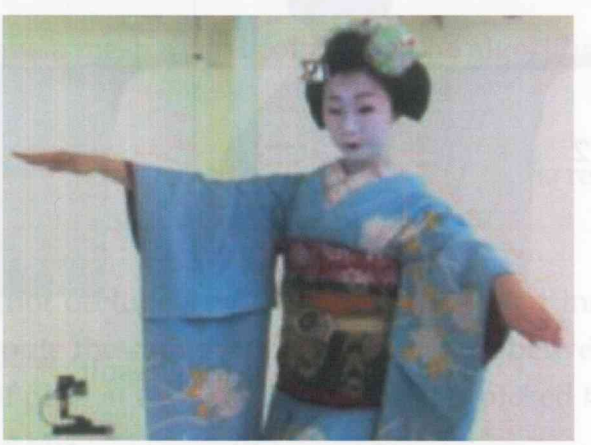


Figure 2.31: Reference image (observed from CAM_{12} in Figure 2.23)

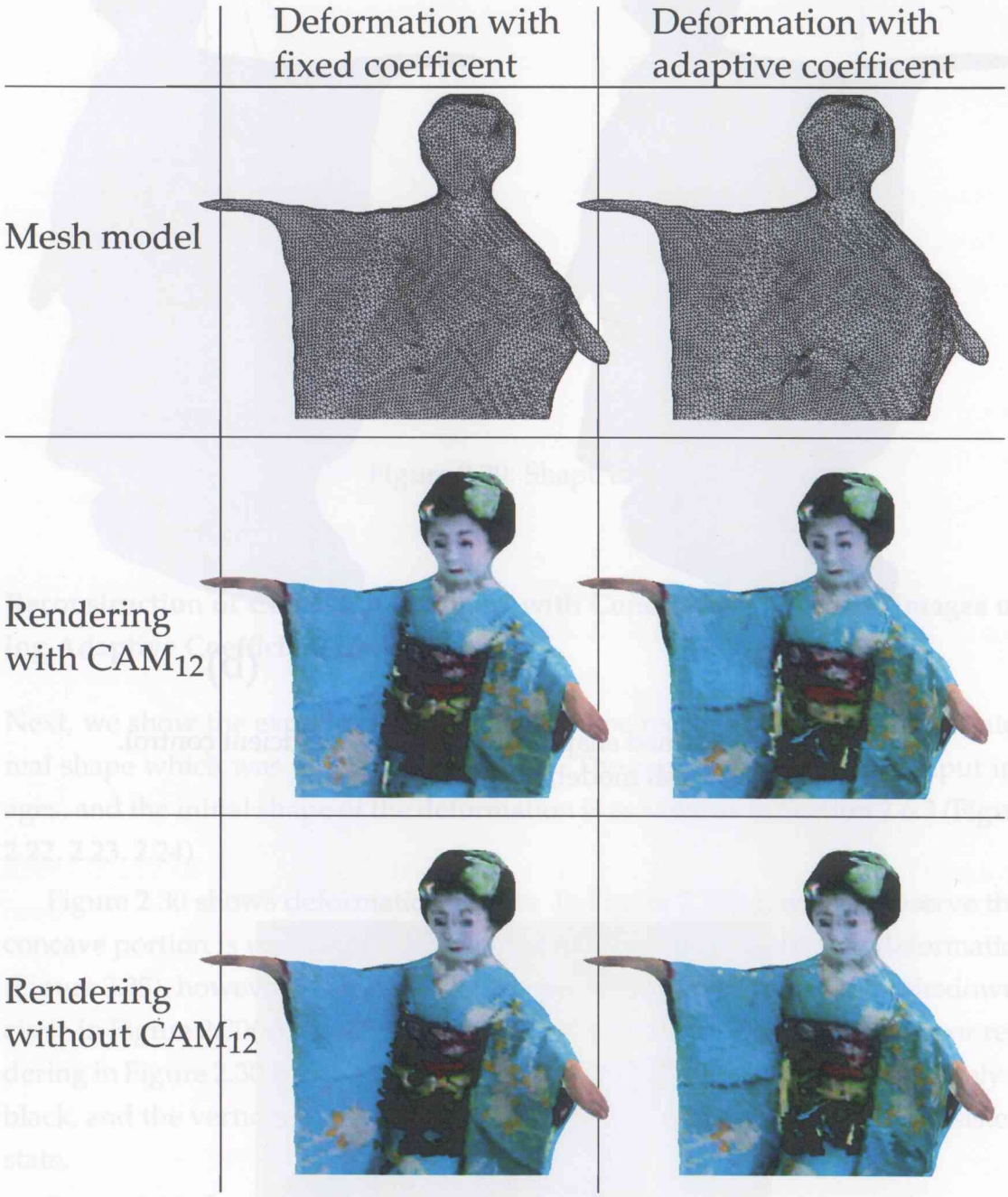


Figure 2.32: Rendering result

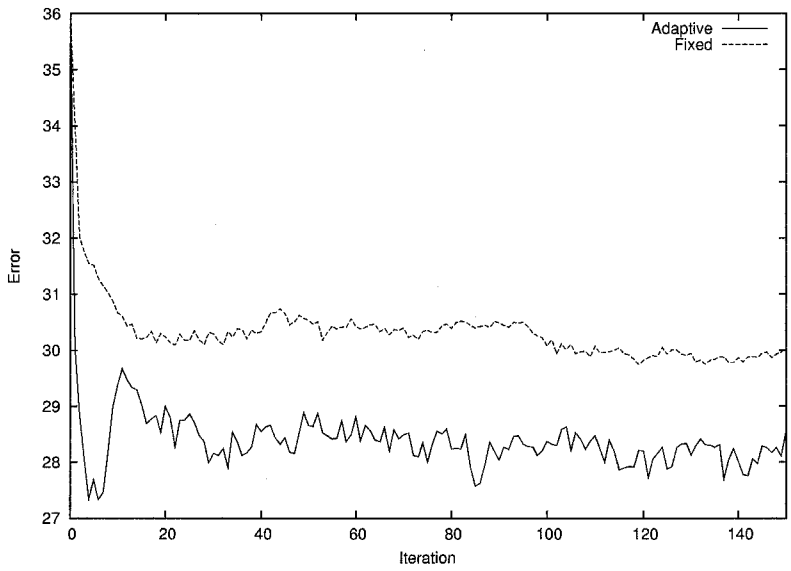


Figure 2.33: Rendering error

without observed image at CAM_{12} which is equal to the rendering viewpoint. We can observe that adaptive coefficient control produces more accurate shape (Figure 2.32, top row) and gives better rendering results both with and without CAM_{12} (Figure 2.32, middle and bottom row) qualitatively.

Figure 2.33 shows quantitative evaluation. We define the rendering error as follows:

1. Suppose we have a mesh model colored by the algorithm described in Section 2.6.3.

2. For each vertex v of the mesh model, project it onto the rendering viewpoint (CAM_{12}). We denote the project position by v'.

3. If v is visible (not occluded by others and projected inside of the camera window), compute the sum of absolute difference between the vertex color and the color of pixel at v'. Note that the pixel is picked from the real image observed by CAM_{12}.

4. Let the rendering error be the average of the sum of absolute difference above.

From this graph, we can observe that adaptive coefficient control gives better result than fixed coefficient deformation.

2.8 Summary

We introduced a static 3D shape estimation method from multi-viewpoint images. We employed simple iterative computation in SNAKE manner, which requires to start its deformation process with a shape entirely covering / covered by the object and shrink / expand until it stops. We can use the visual hull of the object as an initial shape of shrinking, since the visual hull assures the real shape to be encaged in it.

Compared with the Space-Carving method, which employs photometric consistency as its main reconstruction cue, our approach additionally employs geometric continuity and a silhouette constraint. Such rich constraints make our approach more stable and accurate. Moreover, our deformable mesh model can be extended to dynamic inter-frame deformation, which will enable us to analyze dynamic object motion and realize highly efficient data compression. The next chapter describes this inter-frame deformation algorithm.

This intra-frame deformation can be used for key-frame generation of 3D video as described in Section 1.3.1.

Chapter 3

Deformable Mesh Model for Dynamic 3D Shape Estimation

In this chapter, we introduce the inter-frame deformation using our deformable mesh model. The purpose of this deformation is to estimate the object shape and motion between two frames.

In recent years, many studies have been done on 3D shape and motion reconstruction. For static 3D shape reconstruction, several frameworks combining multiple cues such as photometric stereo or silhouette were proposed to accomplish better stability and accuracy as described in Chapter 2. For 3D motion recovery, Heap[HH96] proposed human hand tracking from camera images using a given deformable hand model. Bottino[BL01] tracked 3D human action from multi-viewpoint silhouettes with a known object model. Vedula[VBR+99] introduced a framework to compute dense 3D motion flow from 2D optical flows with / without object shape.

The problem we consider in this chapter is how we can estimate *dynamic* 3D shape from multi-viewpoint video, i.e., time-varying images. That is, we focus on how to estimate the shape and motion of the object *simultaneously*. A naive method for this shape and motion estimation problem would not be simultaneous:

Step 1. reconstruct 3D shape for each frame,

Step 2. estimate 3D motion by establishing correspondences between a pair of 3D shapes at frames t and $t+1$.

However, this approach consists of two-stage computational model and it is not

so easy to manage each stage to cooperate with the other. This is because that meshes representing object shape at each frame can

- have own *global* and *local* topology, and
- consist of different number of vertices.

Here, global mesh topology means mathematical topology of mesh, i.e., geometric genus of mesh. Figure 3.1 shows examples of mesh models in different global topology:

- The left side of the top row shows a genus-0 mesh representing a person opening her arms and legs. It is topologically equivalent to sphere (right) from mathematical point of view.
- The middle row shows two genus-1 meshes (left and center) which are topologically equivalent to ring-torus (or roughly speaking, *donut*) on the right. Note that left and center meshes are topologically equivalent while the left one has a hole between its legs and the center one has under its arm.
- The bottom row shows two genus-2 meshes (left and center) which are topologically equivalent to double-torus (or a letter of *eight*) on the right.

On the other hand, local mesh topology is the local connectivity of a vertex of the mesh model. In other words, the number of vertices neighboring to a vertex. It is intuitively difficult to establish dense, vertex-wise correspondences between two meshes with different genus. Furthermore, even if two meshes have same genus, it is also difficult to find matching between two meshes which have different number of vertices and local connectivity.

For these reasons, we believe that a unified computational model, i.e., simultaneous recovery of 3D shape and motion, is better than the two-stage approach. As illustrated in Section 1.3.1 and Figure 1.4, the outline of our algorithm is as follows.

1. Suppose we have an initial mesh M_t representing the object shape at a frame t.
2. To estimate the shape and motion at the next frame, deform M_t so as to satisfy constraints which should be fulfilled by the object shape at frame $t+1$.

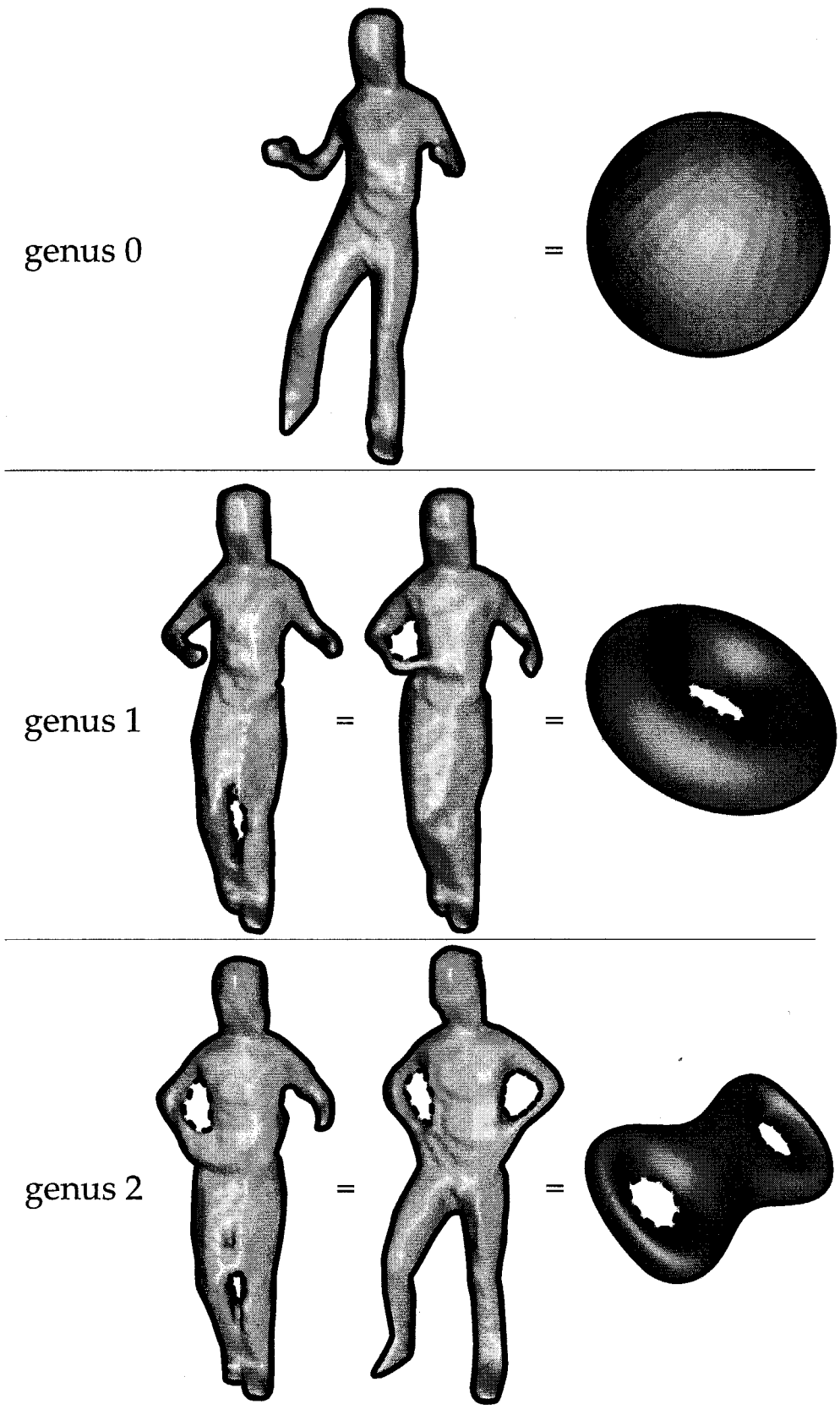


Figure 3.1: Genus of mesh

3. Let deformed M_t be M_{t+1}, the object shape at $t+1$.

4. Since we deformed M_t to M_{t+1}, we have per-vertex correspondences between them, i.e., the motion from t to $t+1$.

5. To obtain M_{t+2}, go back to 2. and use M_{t+1} as the initial shape of the deformation.

Note that "mesh deformation" means translations of vertices alone. As described in Section 2.7, our "deformation" only performs vertex translation, not addition / collapse of vertex to enable inter-frame compression of mesh data[BSM+03][IR03].

Toward the simultaneous recovery, for example, Vedula[VBSK00] showed an algorithm to recover shapes represented by voxels in two frames and per-voxel-correspondence between them simultaneously. Plänkers[PF03] proposed a method which uses a soft object model given a priori and refine it based on photometric and silhouette constraints. Compared with these algorithms, our algorithm has following advantages:

- it can cope with global topology change of the object, and
- it does not require a model given a priori.

In this chapter, we describe basics of our deformable mesh model for dynamic shape estimation, and show how we deal with global topology change in the next chapter.

3.1 Problem Description

The problem we consider is 3D shape and motion estimation of the object between two successive frames. We assume the object which has

- arbitrary but smooth and continuous shape,
- arbitrary motion,
- Lambertian surface reflectance, and
- constant global topology.

Here constant global topology means that the genus of the object in two frames are equivalent.

We represent the object shape by deformable mesh model, and the object motion by translations of vertices composing the mesh. The input data of our algorithm are:

- multi-viewpoint images and object silhouettes at both frame t and $t+1$, and
- a mesh model representing the object shape at frame t.

The output data are:

- the object shape at frame $t+1$ represented by a mesh model, and
- the object motion from t to $t+1$ represented as translations of vertices.

3.2 Approach

As described above, we employ deformation based approach to estimate the object shape and motion simultaneously. The most important difference between the intra-frame deformation in Chapter 2 and the inter-frame deformation in this chapter is that we cannot start the deformation with a mesh which ensures that the object is encaged in it.

That is, in the intra-frame deformation, we used the visual hull of the object as the initial shape of the deformation. Since the visual hull ensures that it encages the real object shape, we could take a simple strategy in deformation in SNAKE manner – shrink if photometric or silhouette preserving force do not work. This shrinking strategy was implemented into the internal force which shrinks the mesh while it keep the mesh locally smooth (Section 2.4.2, Equation (2.8)).

On the other hand, in the inter-frame deformation, we do not have such a reasonable initial shape and basic deformation strategy. This is because:

- we have to start our deformation with the mesh representing the object shape at t, and
- we have no a priori knowledge about location to which the mesh basically deform.

To cope with this problem, Vedula[VBSK00] proposed a kind of brute-force, reconstruction-oriented approach which searches all solution space, that is, 6D correspondence space defined as the direct product of 3D space at t and $t+1$. This approach depends on the spacio temporal photo-consistency on every object surface, but we assume that we cannot expect that every vertices have prominent texture. On the other hand, Plänkers[PF03] proposed a model-driven, tracking-oriented approach using a priori soft object model and track the object motion with 2.5D depth map first, and then deform the soft object to fit the object outline. This approach requires object models designed a priori for each target, but it is not so easy to prepare models for persons with various cloth, e.g., Figure 2.19 and Figure 2.22.

Compared with these approaches, we propose to introduce basic deformation strategy for the inter-frame deformation by roughly estimating the object motion instead of shrinking strategy in Chapter 2. To estimate the object motion, we use the visual hull at frame $t+1$ given by input images. By adding this estimated motion as new constraint, we extend our deformable mesh model in Chapter 2 to be able to deform frame by frame.

In the following sections, we describe how we modify and add the constraints which should be fulfilled by the object at $t+1$ and how we compute the deformation process.

3.3 Constraints

3.3.1 Photometric constraint

> *A patch in the mesh model should be placed so that its texture, which is computed by projecting the patch onto a captured image of visible viewpoint, should be consistent irrespectively of onto which image and frame it is projected.*

Since we focus on not shape estimation on two frames, but shape and motion estimation, it is not sufficient that textures of a patch are consistent at frame t and also at $t+1$ individually. Such constraint allow a patch which represents different part of the object surface at t and $t+1$. To estimate object shape and motion, we need a vertex such that its texture is also consistent between frame t and $t+1$.

Note that since we assumed that we start the deformation with a shape that represents the object shape at frame t, visible cameras and position of each vertex

at frame t are fixed through the deformation process. Visible cameras at frame $t+1$ will change in deformation, and may be different from that of t.

3.3.2 Silhouette constraint

> *When the mesh model is projected onto an image plane, its 2D silhouette should be coincide with the observed object silhouette at frame $t+1$ on that image plane. The contour generators of each projected mesh silhouette should be located contiguously on the mesh surface.*
>
> *Moreover, the contour generators at frame $t+1$ should be equal to or nearby the contour generators at frame t.*

In the inter-frame deformation, we have two constraints about the silhouettes. First, as same as that of the intra-frame deformation (Section 2.3.2), projected mesh silhouette should

1. match with the observed silhouette, and
2. be generated by vertices located contiguously on 3D surface.

Next, as we start the deformation with a mesh representing the object shape at frame t, we know vertices such that each of them was a part of contour generators at t. By assuming the object surface to be smooth, we can expect that the contour generators at $t+1$ will located nearby those at t.

This constraint states that

1. apparent contours on viewpoints at $t+1$, and
2. spacio-temporal continuity of contour generators

should be satisfied.

3.3.3 Smoothness constraint

> *The 3D mesh should be locally smooth and should not intersect with itself.*

By assuming the surface to be smooth, we make the mesh model to be able to interpolate its shape especially where no cameras can observe a vertex.

3.3.4 Motion flow constraint

A mesh vertex should drift in the direction of the motion flow of its vicinity.

In the intra-frame deformation, we could use shrinking strategy when a vertex has no appropriate destination to deform. On the other hand, we cannot have such an implicit strategy since we do not start the deformation from some special condition like the intra-frame deformation. Furthermore, we cannot expect that photometric constraint can lead all the vertices from t to the object surface at $t+1$ because:

- the photo-consistency function $E_e(v)$ (defined below) could not find a point on the object surface for all vertices since we cannot expect that the object surface has prominent textures all over it, and
- we cannot define the photo-consistency function $E_e(v)$ as a strictly increasing or decreasing function even if the object surface has prominent textures everywhere on it. So the photo-consistency function $E_e(v)$ may have local optimas.

So we need a force to guide the deformation process in case if photometric force cannot find an appropriate deformation destination. Here we introduce "motion flow" which is a roughly estimated flow of object motion from visual hulls at two frames t and $t+1$, and let the mesh model to deform according to this motion flow.

3.3.5 Inertia constraint

The motion of a vertex should be temporally smooth and continuous.

The motion flow constraint utilize a result of motion estimation between t and $t+1$. Similarly to the motion flow constraint, this inertia constraint states that object motion estimated between $t-1$ and t can be used to guide the deformation process by assuming the object motion to be smooth and continuous.

3.4 Motion Estimation

In this section, we introduce two types of motion estimation. One is between frame t and $t+1$, and the other is between $t-1$ and t. These two estimation seem

to be redundant especially if the object motion is simple and global topology of the object is kept to be genus-0. However, when global topology of the object change through frames, which will be described in Chapter 5, we cannot expect that we can estimate the object motion with both approach. So to cope with time-varying global object topology, we introduce two different approaches to make the estimation more stable.

This is a mutual compensation of estimation cues as described in Section 1.3.

3.4.1 Motion Estimation from Two Visual Hulls

As described above, we assume that we have silhouette images and visual hulls at each frame. With these visual hulls, i.e., sets of voxels, we compute rough correspondences between them by simple point-set-deformation algorithm[Bur81].

Suppose we have two point sets A and B,

$$\begin{aligned} A &= \{\boldsymbol{a}_i \mid i = 1, \ldots, N_A\}, \\ B &= \{\boldsymbol{b}_j \mid j = 1, \ldots, N_B\}, \end{aligned} \tag{3.1}$$

where $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$ denote i-th and j-th point of A and B respectively, N_A and N_B denote the number of points in each set.

Let $d(\boldsymbol{x}, \boldsymbol{x}')$ denote the Euclidean distance between a point $\boldsymbol{x}$ and $\boldsymbol{x}'$. With this distance function d, we can define a point in a point set Y which is closest from $\boldsymbol{x}$. We denote such a point by $\boldsymbol{y}_{i_x}$, where

$$i_x = \underset{i=1,\ldots,N_Y}{\operatorname{argmin}} d(\boldsymbol{x}, \boldsymbol{y}_i), \tag{3.2}$$

and N_Y denotes the number of points in Y, and $\boldsymbol{y}_i$ the i-th point in Y. So the nearest point displacement vector from $\boldsymbol{a}_i$ to B can be defined as follows:

$$D(\boldsymbol{a}_i, B) = \boldsymbol{b}_{j_{a_i}} - \boldsymbol{a}_i, \tag{3.3}$$

and similarly, $\boldsymbol{b}_j$ to A as

$$D(\boldsymbol{b}_j, A) = \boldsymbol{a}_{i_{b_j}} - \boldsymbol{b}_j. \tag{3.4}$$

If these displacement vectors from A to B and B to A were bijective, we may use them as correspondences between A and B. In general, however, since N_A and N_B, the number of points in each set is not equal, and/or point arrangements of A and B are different, the displacement vectors may be inconsistent.

To obtain consistent displacement vectors, we define smoothed displacement vector from $\boldsymbol{a}_i$ to B as follows:

$$D(\boldsymbol{a}_i) = \frac{\sum_{i'=1}^{N_A} G_A(\boldsymbol{a}_i, \boldsymbol{a}_{i'}) D(\boldsymbol{a}_i, B)}{\sum_{i'=1}^{N_A} G_A(\boldsymbol{a}_i, \boldsymbol{a}_{i'})} - \frac{\sum_{j'=1}^{N_B} G_B(\boldsymbol{a}_i, \boldsymbol{b}_{j'}) D(\boldsymbol{b}_{j'}, B)}{\sum_{j'=1}^{N_B} G_B(\boldsymbol{a}_i, \boldsymbol{b}_{j'})}, \tag{3.5}$$

where $G_A(\boldsymbol{a}_i, \boldsymbol{a}_{i'})$ and $G_B(\boldsymbol{b}_j, \boldsymbol{b}_{j'})$ denote Gaussian functions with smoothing radius σ,

$$\begin{aligned} G_A(\boldsymbol{a}_i, \boldsymbol{a}_{i'}) &= \exp\left(-\frac{|\boldsymbol{a}_i - \boldsymbol{a}_{i'}|^2}{\sigma^2}\right), \\ G_B(\boldsymbol{a}_i, \boldsymbol{b}_{j'}) &= \exp\left(-\frac{|\boldsymbol{a}_i - \boldsymbol{b}_{j'} - D(\boldsymbol{b}_{j'}, A)|^2}{\sigma^2}\right). \end{aligned} \tag{3.6}$$

Note that G_A and G_B are asymmetric to let one displacement vector pull the points while the other push it.

This $D(\boldsymbol{a}_i)$ pushes each point in A toward B. So we can obtain correspondences between A and B by iteratively moving the points in A to B according to $D(\boldsymbol{a}_i)$ until A will be close enough to B.

Step 1. Let A and B be point sets.

Step 2. Compute the displacement vector $D(\boldsymbol{a}_i)$ for each point in A.

Step 3. Move $\boldsymbol{a}_i$ slightly according to $D(\boldsymbol{a}_i)$.

Step 4. If $\max_{\boldsymbol{a}_i \in A} D(\boldsymbol{a}_i, B)$ is not small enough, go back to Step 2.

Step 5. Finally, we have deformed point set A which is close to B.

With this algorithm, we can compute correspondences between voxels of the visual hull at frame t and $t+1$:

Step 1. Compute two visual hulls V_t at t and V_{t+1} at $t+1$ from multi-viewpoint silhouettes at both frame.

Step 2. Obtain boundary voxels from V_t and V_{t+1}, and let them be A and B respectively.

Step 3. Compute correspondences between A and B by the algorithm described above.

Here, boundary voxels are a set of voxels in a visual hull such that at least one of its neighbors is vacant.

Use of Euclidean distance implies that corresponding candidate is determined individually by single point-to-point distance, and local connectivities between vertices are implicitly implemented by Gaussian smoothing in Equation 3.5. This is simple and reasonable approach if the arrangement of two point sets are similar to each other. We discuss about rough motion estimation for topologically changing deformation in the next chapter.

3.4.2 Motion Estimation as Inertia

The motion estimation described above was an estimation using two frames t and $t+1$. Here we introduce another estimation as extrapolation of vertex positions at $t+1$ using the mesh models at t and $t-1$.

Suppose we have started our deformation at frame $t-1$ with a mesh M_{t-1} and it has been already deformed to be the object shape at t. Let this deformed mesh be denoted by M_t. Now, we are about to deform M_t so as to represent the object shape at $t+1$.

Since we have defined our deformation process as translations of vertices, we have per-vertex correspondences between two meshes M_t and M_{t-1}. Let us denote the position of a vertex v at frame t and $t-1$ by $\boldsymbol{q}_v^t$ and $\boldsymbol{q}_v^{t-1}$ respectively. We estimate the position of v at $t+1$ by linear extrapolation,

$$\begin{aligned} \boldsymbol{q}_v^{t+1} &= \boldsymbol{q}_v^t + \left(\boldsymbol{q}_v^t - \boldsymbol{q}_v^{t-1}\right) \\ &= 2\boldsymbol{q}_v^t - \boldsymbol{q}_v^{t-1}. \end{aligned} \tag{3.7}$$

Note that the time-step between $t-1$, t, and $t+1$ is constant. With this extrapolation, we obtain the estimated positions for each vertex.

3.5 Forces on a Vertex

In this section, we introduce forces which represent reconstruction constraints. Note that we assumed that the deformation starts with a shape that represents the object shape at frame t. We denote the initial position of a vertex v, i.e., the position of v at frame t by $\boldsymbol{q}_{\hat{v}}$.

3.5.1 Photometric Force

To obtain textures of v at frame t, we have to note that:

- frame t is the initial point of deformation,
- the position of v at t is given as $q_{\hat{v}}$, and
- $C_{\hat{v}}$, visible cameras of v at t, is fixed through the deformation process.

So textures at frame t is also fixed through the deformation process and given by projecting a vertex v at $q_{\hat{v}}$ onto viewpoints which can observe v at t.

Textures at frame $t+1$ is given by similar to the intra-frame deformation. Once the deformation has started, the position of vertex v is considered as the position at frame $t+1$. We can obtain textures at $t+1$ by projecting v onto viewpoints from which v is visible at $t+1$. In what follows, we omit the superfix $t+1$ for a vertex, visible cameras of that vertex, and all others concerning frame $t+1$ for brevity, and we denote x at frame t as $\hat{x}$. For example, C_v is visible cameras of v at frame $t+1$ and $C_{\hat{v}}$ is that of frame t.

Photometric constraint states that these textures at two frames should be consistent. So we define the photometric force for the inter-frame deformation as follows:

$$\boldsymbol{F}_e(v) \equiv \begin{cases} \nabla E_e(\boldsymbol{q}_v, \boldsymbol{q}_{\hat{v}}), & \text{if } N(C_v) \geq 2 \text{ and } N(C_{\hat{v}}) \geq 2, \\ 0, & \text{otherwise,} \end{cases} \tag{3.8}$$

where $N(C_v)$ denotes the number of cameras in C_v, $E_e(\boldsymbol{q}_v, \boldsymbol{q}_{\hat{v}})$ the correlation of textures to be mapped around v:

$$E_e(\boldsymbol{q}_v, \boldsymbol{q}_{\hat{v}}) \equiv \frac{1}{N(C_v) + N(C_{\hat{v}}) - 1} \left(\sum_{c \in C_v \setminus c_m} \mathrm{NCC}\,(c, c_m) + \sum_{\hat{c} \in C_{\hat{v}} \setminus \hat{c}_m} \mathrm{NCC}\,(\hat{c}, \hat{c}_m) \right), \tag{3.9}$$

where c_m and $\hat{c}_m$ denote the most facing camera in C_v and $C_{\hat{v}}$ respectively, c a camera in C_v except c_m, $\hat{c}$ a camera in $C_{\hat{v}}$ except $\hat{c}_m$, and $\mathrm{NCC}\,(c, c_m)$ the normalized cross correlation function defined in Equation 2.3.

3.5.2 Silhouette Preserving Force

Let us start the definition of the silhouette preserving force for the inter-frame deformation from that of the intra-frame deformation in Section 2.4.2.

To make the mesh model to preserve the observed object outlines at viewpoints, we first replace definition of partial silhouette preserving force $f_s(v)$ in Section 2.4.2 so that it preserves the object outlines not at t but $t+1$. We reuse the symbol $f_s(v)$ to denote this modified force.

Next, to realize temporal continuity of contour generators, we modify the definition of the local support of silhouette force. Let us recall that the local support of each vertex was computed with consideration to others nearby the vertex and its competitors in Section 2.4.2. In addition to these two factors, we introduce temporal continuity factor as follows.

1. At the beginning of the deformation, that is, at frame t, find vertices such that they generates apparent contours of each viewpoint. Then, for each such vertex, set its likelihood defined as Section 2.4.2. We denote this likelihood of a vertex v at frame t by $\hat{L}(v)$. Note that:
   (a) this step is done only at once, and $\hat{L}(v)$ is fixed through the deformation, and
   (b) the value of $\hat{L}(v)$ is 0 if v is not a part of contour generator at frame t.
2. Change the initial value of likelihood to $\hat{L}(v)$. In Section 2.4.2, we initialized the likelihood to 0 at every start of iteration evenly, however, to make vertices which were contour generators at frame t to take higher likelihood, we use $\hat{L}(v)$ as the initial value.

Applying these modifications, the likelihood $L(v)$ is computed as follows:

- At the beginning of the deformation, compute $\hat{L}(v)$ for each vertex.
- At every iteration,

  Step 1. Initialize the likelihood of each vertex to $\hat{L}(v)$.

  Step 2. For each point of silhouette outlines, let each vertex accumulate its likelihood of contour generator. We add the likelihood by the following rule:
  - if a silhouette point has n corresponding vertices, add $\frac{1}{n}$ to each vertex likelihood.

  We denote the competitors of v by $\mathrm{Comp}(v)$.

  Step 3. For each vertex, summing up the likelihood within its local support. We denote accumulated value by $S_L(v)$.

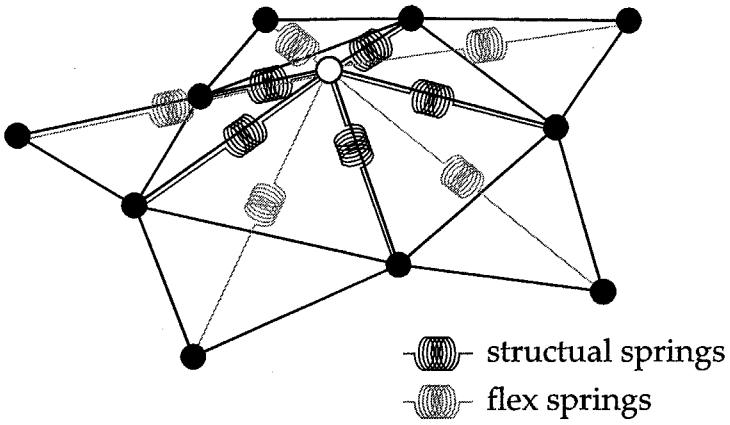


Figure 3.2: Spring model

Step 4. For each vertex, compute the modified local support:

$$L(v) = \frac{S_L(v)}{S_L(v) + \sum_{v' \in \mathrm{Comp}(v)} S_L(v')}, \tag{3.10}$$

and total silhouette preserving force:

$$\boldsymbol{F}_s(v) \equiv L(v) \sum_{c \in C_v} f_s(v, c). \tag{3.11}$$

3.5.3 Internal Force

As described in Section 2.4.2, we cannot use simple internal force used in the intra-frame deformation (Equation 2.8). Equation 2.8 makes the mesh shrink while it keeps it smooth. It was appropriate because the intra-frame deformation starts with visual hull which encages the object, and shrinking was reasonable strategy to find the real object surface. On the other hand, in the inter-frame deformation now we consider, it starts its deformation process with the shape at frame t, and change the mesh so as to fit the shape at $t+1$. So we should not make the mesh to shrink.

To realize smoothing function solely, we employ spring model. As shown Figure 3.2, internal force on a vertex v consists of two components:

- Structural springs connecting to the neighboring vertices of v.
- Flex springs connecting to the vertices $\check{v}_j$ such that v and $\check{v}^j$ is a diagonal vertex of two neighboring triangle patches.

Structural spring keeps neighboring vertices in a certain length, and flex spring prevents folding of mesh.

With spring constants $k_s(v, v_j)$ and $k_f(v, v_j)$, we define the internal force on v as follows:

$$\boldsymbol{F}_i(v) = \sum_{j=1}^{N} \boldsymbol{f}_i(v, v_j, k_s(v, v_j)) + \sum_{j=1}^{N} \boldsymbol{f}_i(v, \check{v}_j, k_f(v, \check{v}_j)) - \dot{\boldsymbol{q}}_v, \tag{3.12}$$

where v_j denotes the j-th neighboring vertex, $\check{v}_j$ the j-th diagonal vertex, N the number of neighboring vertices, $k_s(\cdot)$ the structural spring constant between two vertices, $k_f(\cdot)$ the flex spring constant, and $\dot{\boldsymbol{q}}_v$ the damping force of spring proportion to the velocity of v. $\boldsymbol{f}_i(\cdot)$ is Hookean spring force given by

$$\boldsymbol{f}_i(v_a, v_b, k) = k \frac{\|\boldsymbol{q}_{v_a} - \boldsymbol{q}_{v_b}\| - l(v_a, v_b)}{\|\boldsymbol{q}_{v_a} - \boldsymbol{q}_{v_b}\|} (\boldsymbol{q}_{v_a} - \boldsymbol{q}_{v_b}), \tag{3.13}$$

where $l(v_a, v_b)$ denote the nominal length of the spring between v_a and v_b. Note that number of diagonal vertices is equal to that of neighboring vertices, N.

3.5.4 Drift Force

As described in Section 3.4.1, we have roughly estimated motion flow as voxel-wise correspondences between t and $t+1$ since we assume that we have multi-viewpoint silhouette images and a visual hull for each frame.

Let us denote the voxel set at t by V_t, and the voxel set at $t+1$ by V_{t+1}. Then we represent voxel-wise correspondences by a set of *correspondence lines*:

$$L_t = \left\{ l^i \middle| \, i = 1, \ldots, N(V_t) \right\}, \tag{3.14}$$

where l^i denotes the correspondence line starting from ith voxel in V_t and $N(V_t)$ the number of voxels in V_t.

Once the motion flow is obtained, we define the potential field $E_d(v)$ generated by this flow. First, let l_v denote the correspondence line in L_t closest to v, $\boldsymbol{p}_{l_v,v}$ the point on l_v closest to v, and $\boldsymbol{s}_{l_v}$ the starting point of the correspondence line l_v. Then, we define the potential field as a function of the distance from v_t to l_{v_t} and the distance from $\boldsymbol{s}_{l_{v_t}}$ to $\boldsymbol{p}_{l_{v_t},v_t}$:

$$E_d(\boldsymbol{q}_v) \equiv \|\boldsymbol{s}_{l_v} - \boldsymbol{p}_{l_v,v}\|^2 - \|\boldsymbol{q}_v - \boldsymbol{p}_{l_v,v}\|^2. \tag{3.15}$$

Finally, we define the drift force $\boldsymbol{F}_d(v)$ at vertex v as the gradient vector of $E_d(\boldsymbol{q}_v)$:

$$\boldsymbol{F}_d(v) \equiv \nabla E_d(\boldsymbol{q}_v). \tag{3.16}$$

3.5.5 Inertia Force

If we assume that the interval between successive frames is short enough, we can expect that the motion of the object to be smooth and continuous. This assumption tells us that we can predict the location of a vertex at $t+1$ from its motion history as described in Section 3.4.2.

We can represent such predictions as a set of prediction lines connecting $\boldsymbol{q}_{v_t}$ and $\hat{\boldsymbol{q}}_v$, where $\hat{\boldsymbol{q}}_v$ denotes the predicted location of v. Then we can define the inertia force $\boldsymbol{F}_n(v)$ in just the same way as the drift force $\boldsymbol{F}_d(v)$:

$$\boldsymbol{F}_n(v) \equiv \nabla E_n(\boldsymbol{q}_v), \tag{3.17}$$

where $E_n(\boldsymbol{q}_v)$ denotes the potential field defined for the set of prediction lines, defined in just the same way as in equation (3.15).

3.5.6 Overall Vertex Force

Finally we define the vertex force $\boldsymbol{F}(v_{t+1})$ with coefficients $\alpha, \beta, \gamma, \delta, \epsilon$, and ζ as follows:

$$\boldsymbol{F}(v) \equiv \alpha\boldsymbol{F}_i(v) + \beta\boldsymbol{F}_e(v) + \gamma\boldsymbol{F}_s(v) + \delta\boldsymbol{F}_d(v) + \epsilon\boldsymbol{F}_n(v). \tag{3.18}$$

3.6 Computation Algorithm

3.6.1 Initial Shape

We assumed that we have a mesh model representing the object shape at frame t and deform it to be that of $t+1$, $t+2$, ... and so on. To prepare the initial shape to start the inter-frame deformation, we can use the reconstruction result of the intra-frame deformation in Chapter 2.

3.6.2 Deformation Process

In the intra-frame deformation, we could use shrinking strategy realized by internal force (Equation 2.4.2), and deformation process like SNAKE. On the other hand, we do not have such a reasonable a-priori strategy in the inter-frame deformation, and we changed the internal force to physical spring model with damping force. So we employ usual physics simulation here. That is, we solve Newton's law of motion for each vertex.

For each vertex v, Newton's law of motion gives following equation:

$$\begin{aligned} \boldsymbol{F}(v) &= m_v \boldsymbol{a}(v) \\ &= m_v \ddot{\boldsymbol{q}}_v, \end{aligned} \tag{3.19}$$

where m_v is the mass of v, and $\boldsymbol{a}(v)$ the acceleration of v which is equal to the second derivative of position $\boldsymbol{q}_v$. By assuming $m_v = \text{Const.}$ for every vertex, and including it into constants $\alpha, \ldots, \epsilon$ of $\boldsymbol{F}(v)$, we have

$$\boldsymbol{F}(v) = \ddot{\boldsymbol{q}}_v. \tag{3.20}$$

To solve this second order ordinary differential equations, we use the forward Euler method:

$$\ddot{\boldsymbol{q}}_v(\tau) = \boldsymbol{F}(v, \tau), \tag{3.21}$$

$$\dot{\boldsymbol{q}}_v(\tau + \Delta\tau) = \dot{\boldsymbol{q}}_v(\tau) + \Delta\tau \ddot{\boldsymbol{q}}_v(\tau), \tag{3.22}$$

$$\boldsymbol{q}_v(\tau + \Delta\tau) = \boldsymbol{q}_v(\tau) + \Delta\tau \dot{\boldsymbol{q}}_v(\tau), \tag{3.23}$$

where τ denotes a iteration time, $\Delta\tau$ the iteration step. We start this computation with $\tau = t$ until the movement of each vertex lies under a certain threshold. By regarding the final value of τ as $t + 1$, we obtain the shape of the object at $t + 1$ as the result of deformation, and object motion as deformation loci of vertices.

As is well known, the forward Euler method is simple but not optimal to solve ordinal differential equations in general because we have to choose $\Delta\tau$ carefully to be small enough so that it fulfills Courant-Friedrich-Levy condition. Otherwise, the deformation would become unstable, but this increases iteration step count. To solve this problem, many papers, e.g., on cloth simulation in computer graphics, have utilized another method like backward Euler method (implicit integration), or Runge-Kutta method[BW98]. However, these methods require *fu-*

ture value of $\boldsymbol{F}$. For example, backward Euler method is given as follows:

$$\begin{aligned} \ddot{\boldsymbol{q}}_v(\tau) &= \boldsymbol{F}(v, \tau + \Delta\tau), \\ \dot{\boldsymbol{q}}_v(\tau + \Delta\tau) &= \dot{\boldsymbol{q}}_v(\tau) + \Delta\tau\ddot{\boldsymbol{q}}_v(\tau), \\ \boldsymbol{q}_v(\tau + \Delta\tau) &= \boldsymbol{q}_v(\tau) + \Delta\tau\dot{\boldsymbol{q}}_v(\tau). \end{aligned} \tag{3.24}$$

To compute future $\boldsymbol{F}(v, \tau + \Delta\tau)$ at τ, they use Taylor's theorem,

$$\boldsymbol{F}(v, \tau + \Delta\tau) \approx \boldsymbol{F}(v, \tau) + \frac{\delta \boldsymbol{F}(v, \tau)}{\delta \boldsymbol{q}_v}\Delta\tau + \frac{\delta \boldsymbol{F}(v, \tau)}{\delta \dot{\boldsymbol{q}}_v}\Delta\dot{\boldsymbol{q}}_v. \tag{3.25}$$

Note that $\boldsymbol{F}$ is a function of $\boldsymbol{q}_v$ and $\dot{\boldsymbol{q}}_v$ (Equation 3.18). In physics simulation, these derivatives of $\boldsymbol{F}$ is given analytically or can be estimated numerically. However, our definition of $\boldsymbol{F}$ is not differentiable nor numerically extrapolative, since silhouette preserving force $F_s(v)$ does selective operation (Section 3.5.2).

Hence, we cannot use these methods and we use forward Euler method to solve Equation (3.19). So we define the deformation process as follows.

Step 1 Suppose we have an initial mesh model representing the object shape at frame t. If we have been in successive inter-frame deformation, use the mesh deformed from previous frame $t-1$, otherwise, use the intra-frame deformation (in Chapter 2) to obtain the initial shape.

Step 2 Compute the initial likelihood of contour generator $\hat{L}(v)$ for each vertex (in Section 3.5.2).

Step 3 Estimate the motion flow vector for $\boldsymbol{F}_d(v)$ (in Section 3.5.4).

Step 4 Deform iteratively. For each iteration,

- Step 4.1. Compute forces working at each vertex respectively.
- Step 4.2. Compute velocities of vertices according to Equation (3.22).
- Step 4.3. Update positions of vertices according to Equation (3.23).
- Step 4.4. Terminate if the vertex motions are small enough. Otherwise go back to 5.1 .

Step 5 Take the final shape of the mesh model as the object shape at frame $t+1$.

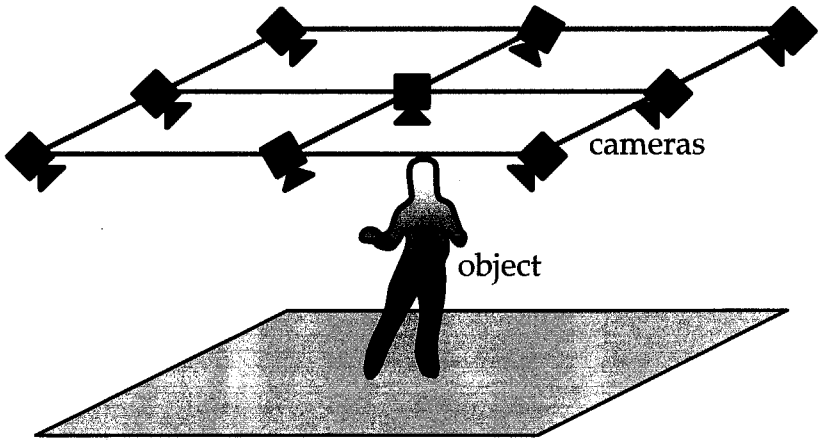


Figure 3.3: Camera arrangement

3.7 Performance Evaluation

Figure 3.3 illustrates camera arrangement in this experiment. We used 9 cameras on the ceiling of the room. Figure 3.4 illustrates estimated motion flows between three successive frames:

- The top row shows captured images from a camera in front of the object.
- The middle row shows the visual hulls of the object which have same global mesh topology.
- The bottom row shows estimated motion flows by algorithm described in Section 3.4.1.

Note that the visual hulls at each frame are good first estimation of the object shape, but they have errors since they only encase the object shapes. For example, we can find phantom volume between their legs. Thus, computed flows on the bottom row and the drift force $F_d(v)$ also contain some errors. However, these errors can be corrected by other forces.

Figure 3.5 and 3.6 illustrate the inter-frame deformation through 3 successive frames. The columns of Figure 3.5 show, from left to right, the captured images, the visual hulls generated by the discrete marching cubes method for each frame, and the mesh models deformed by the inter-frame deformation algorithm proposed above, respectively. Note that captured multi-view video data are not completely synchronized and include motion blur. In this experiments, we used 9 cameras arranged in the same way as Figure 2.14 (a) to capture object images.

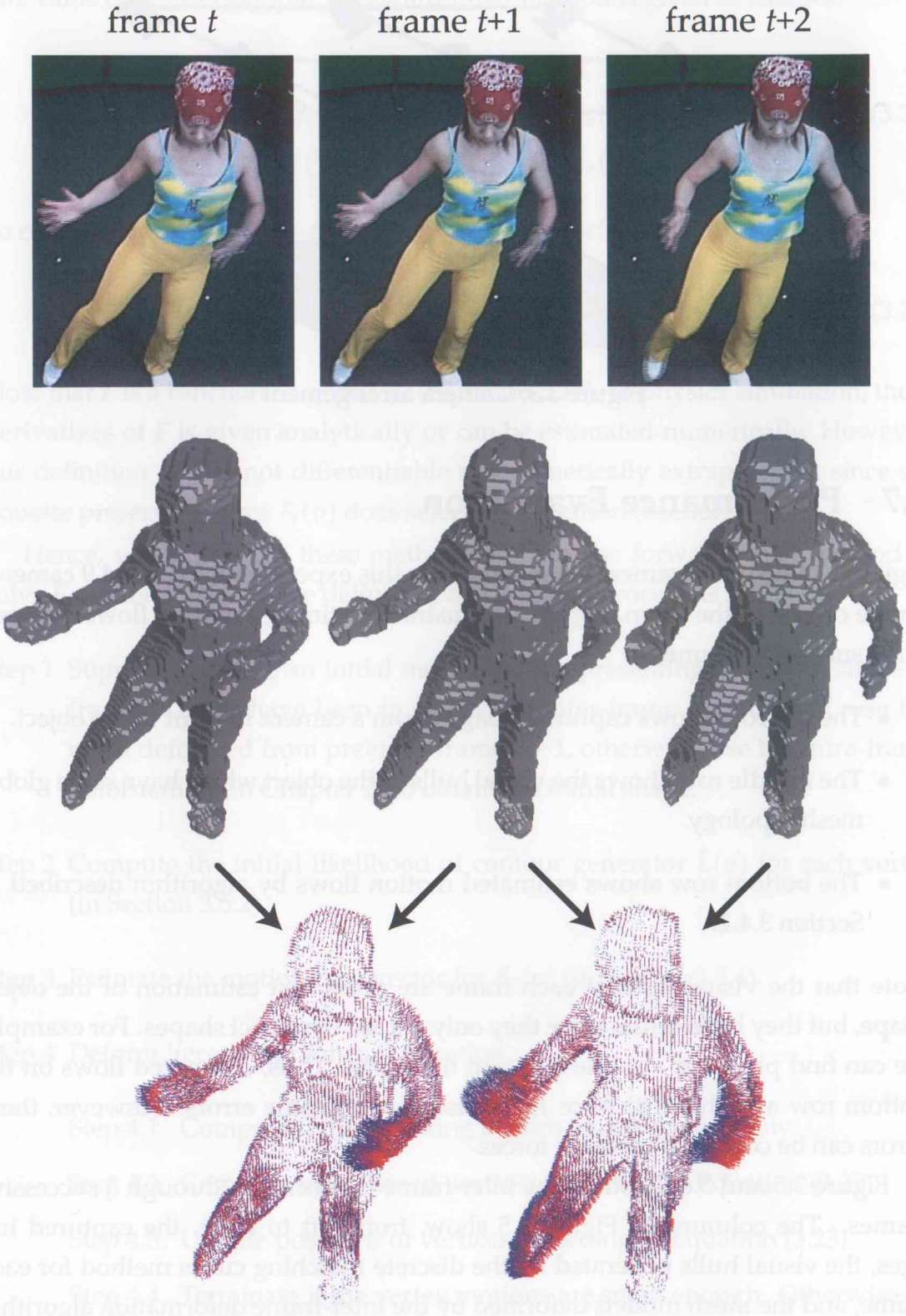


Figure 3.4: Estimated motion flow

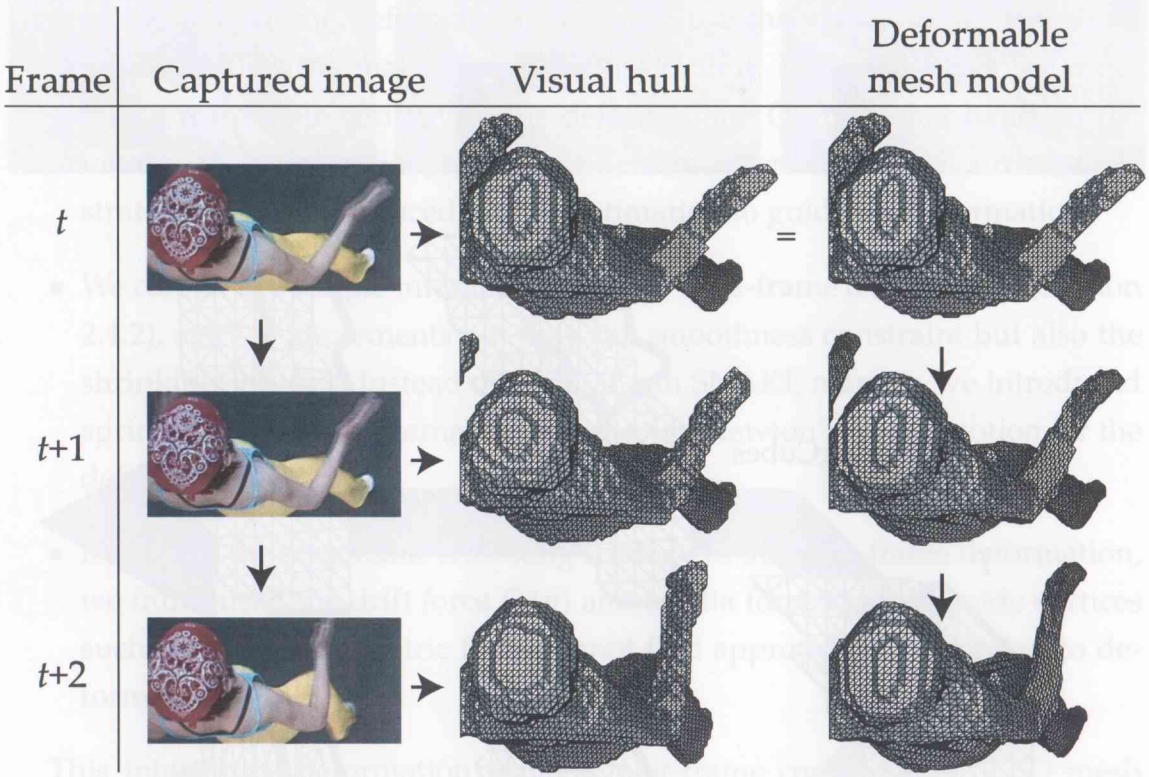


Figure 3.5: Successive deformation results (overview)

The mesh models consist of about 12,000 vertices and 24,000 triangles, and the processing time per frame is about 10 minutes by PC (Xeon 1.7GHz). Note that the visual hull in frame t was used as the initial shape for the intra-frame deformation and then the resultant mesh for the inter-frame deformation. We used fixed coefficients $\alpha = 0.2, \beta = 0.2, \gamma = 0.2, \delta = 0.3, \epsilon = 0.1$ given a priori.

From these results, we can observe:

- Our dynamic mesh model can follow the non-rigid object motion smoothly.
- During its dynamic deformation, our mesh model preserves both global and local topological structure and hence we can find corresponding vertices between any pair of frames for all vertices. Figure 3.6 illustrates this topology preserving characteristic. That is, the left mesh denotes a part of the initial mesh obtained by applying the marching cubes to the visual hull at t. The lower bold arrow stands for the inter-frame deformation process, where any parts of the mesh can be traced over time. Aligned along the upper bold arrow, on the other hand, are parts of the meshes obtained by applying the marching cubes to each visual hull independently, where no vertex

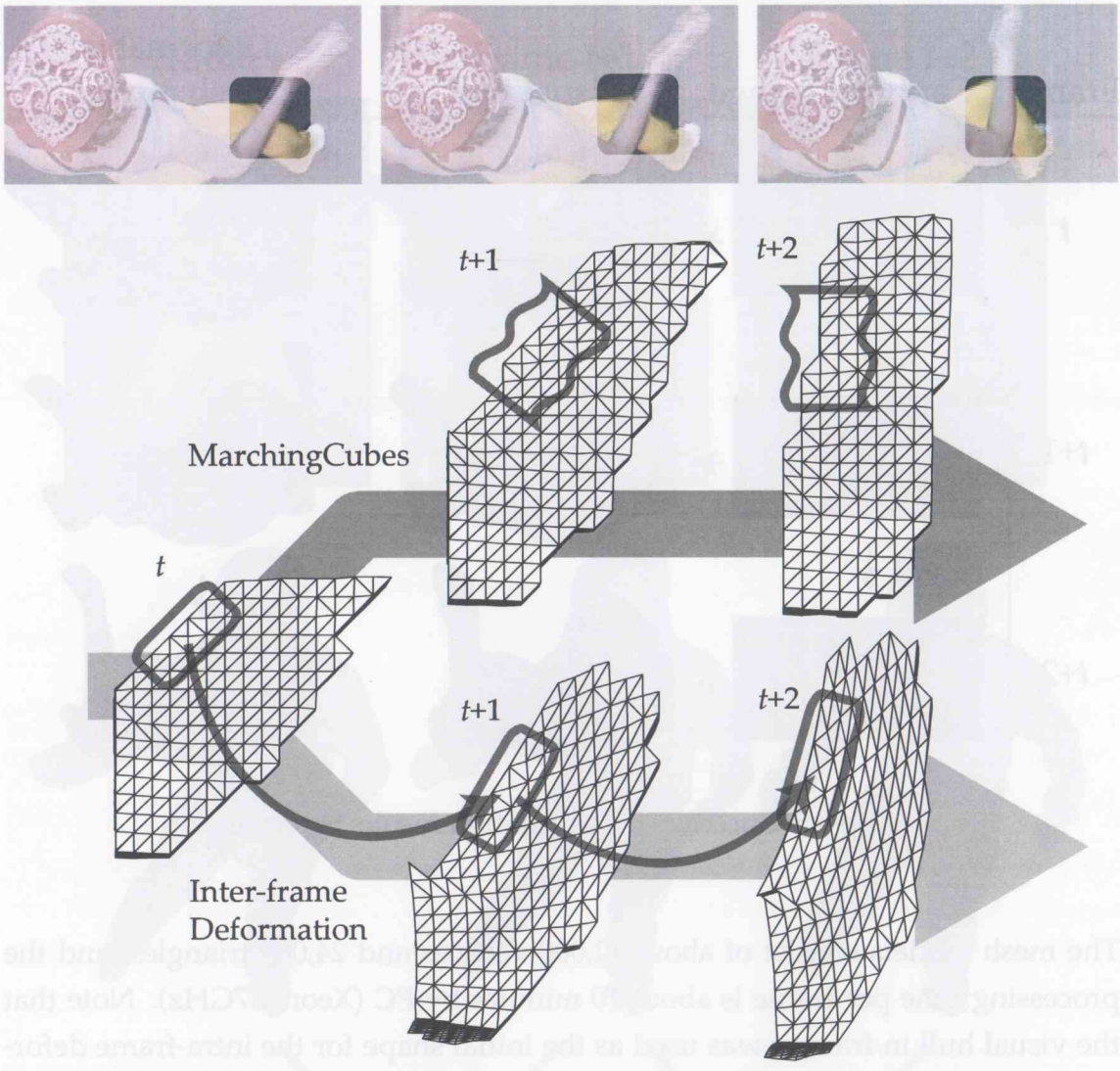


Figure 3.6: Successive deformation results (detailed)

correspondence can be established because the topological structures of the meshes are different.

3.8 Summary

In this chapter, we introduced an algorithm which deforms a mesh model from t to $t+1$. This deformation gives us the object shape at next frame and the object motion between two frames.

Compared with the intra-frame deformation algorithm in Chapter 2, the inter-frame deformation in this chapter is different in the following points:

- In the intra-frame deformation, we could use the visual hull of the object which ensures the real shape of the object in it. So shrinking of the mesh was a reasonable strategy of the deformation. On the other hand, in the inter-frame deformation, we cannot deform according to such a reasonable strategy. So we introduced motion estimation to guide the deformation.

- We cannot use simple internal force in the intra-frame deformation (Section 2.4.2), since it implements not only the smoothness constraint but also the shrinking strategy. Instead of shrinking in SNAKE manner, we introduced spring model as the internal force and solve Newton's law of motion for the deformation.

- Instead of the reasonable shrinking strategy in the intra-frame deformation, we introduced the drift force $F_d(v)$ and inertia force $F_n(v)$ to guide vertices such that the photometric force cannot find appropriate destination to deform.

This inter-frame deformation realizes inter-frame compression of 3D mesh data[BSM+03][IR03], but this sequential computation scheme cannot avoid error accumulation fundamentally. So as described in Section 1.3.1, we need to insert key-frames by the intra-frame deformation introduced in Chapter 2 for error-recovery.

Chapter 4

Heterogeneous Deformation Model for Complex Dynamic 3D Shape Estimation

In Chapter 3, we introduced a basic deformation model to estimate 3D shape and motion from multi-viewpoint images at two frames. In that algorithm, we modeled the object motion as warping. That is, vertices composing object surface can change their position as long as they satisfy their local smoothness constraint defined by spring model (Section 3.5.3). In general, such a loosely constrained model has flexibility to adapt itself to a variety of object shapes as evaluated in Section 2.6.1. Let us call this basic deformation model as *homogeneous* inter-frame deformation model since we managed all the vertices equally.

However, we have already analyzed that vertices can be categorized into two types based on their photometric properties:

- vertices with prominent textures on which the photometric force are dominant, and
- vertices with poor textures on which the internal force are dominant.

That is, characteristics of each vertex are obviously non-uniform. This suggests that we can make simple homogeneous deformation to be more sophisticated by changing the deformation process of each vertex according to their feature. Here, we call such a deformation as *heterogeneous* deformation model.

In this chapter, we propose two types of vertex categorization. One is based on their photometric properties as described above, and the other is based on

their motion. As we proposed in Section 3.4, we have roughly estimated motion vectors. We utilize this motion vectors to categorize vertices into

- vertices in warping, and
- vertices in rigid motion.

These two motion types enable the mesh model to reconstruct the object which consists of different kinds of materials, e.g., rigidly acting body parts and deforming soft clothes or its skins, by a single and unified computational scheme.

In what follows, we introduce how we categorize the vertices, and how we modify the deformation process according to the vertex features. Note that each vertex has not only its position, but also its photometric property and motion type.

4.1 Problem Description

The problem we consider is 3D shape and motion estimation of the object between two successive frames. We assume the object which has

- arbitrary but smooth and continuous shape,
- arbitrary motion, but can be categorized into warping and rigid motion,
- Lambertian surface reflectance, and
- constant global topology.

Here constant global topology means that the genus of the object in two frames are equivalent.

We represent the object shape by deformable mesh model, and the object motion by translations of vertices composing the mesh. The input data of our algorithm are:

- multi-viewpoint images and object silhouettes at both frame t and $t+1$, and
- a mesh model representing the object shape at frame t.

The output data are:

- the object shape at frame $t+1$ represented by a mesh model,

- the object motion from t to $t+1$ represented as translations of vertices,
- motion model of each vertex, and
- texture prominence of each vertex.

4.2 Approach

In general, there is a tradeoff between flexibility and stability. Estimation based on flexible model has possibility to adapt itself to wider variety of targets than that based on restricted model. However, restricted model can achieve more stability while the former may easily affected by noise and break down easily.

In our mesh deformation, "flexible" and "restricted" correspond to per-vertex deformation and group-wise deformation respectively. Per-vertex deformation especially performed by the photometric force has ability to estimate vertex positions precisely, but it is hard to design the photometric force to avoid local optima since textures may have noise or be consistent accidentally by aperture problem. On the other hand, group-wise deformation enables to avoid local minima as long as not most of members have been trapped to, but So in this chapter, we try to find vertices which should be deform as a group to achieve more stability by two approaches.

The first one is a categorization of the object motion into two types:

Restricted part: rigid motion, and

Flexible part: warping,

and modify the deformation process based on these categorization.

The other one is a categorization of vertices if it can lead other vertices in deformation process or not. For each vertex labeled to be a "leader", we modify the deformation process of its neighbors to be led by it.

As a result of modification of the deformation process of each vertex our deformation process turned to be non-uniform. We call this as "heterogeneous deformation". The main topics in this chapter are:

- how we categorize / characterize the vertices in the mesh, and
- how we can utilize such categorization in the deformation process.

Following sections show tow types of categorization described above.

4.3 Vertex Categorization by Clustering Motion Flow Vectors

Since we assumed all the vertices to be warping part in Chapter 3, we first find out vertices which considered to be rigid part by clustering the motion flow estimated in Section 3.4. To estimate vertex-groups in rigid motion, we first cluster the roughly estimated motion flow on the basis of flow vector position and direction. Then for each cluster, we compute its rotation and translation.

We use hierarchical clustering twice to cluster the flow vectors. Hierarchical clustering is a classical algorithm defined as follows.

1. Suppose we have N items to be clustered.

2. Compute an $N \times N$ distance matrix between each item. For example, (i, j) value of this distance matrix stores the distance between i-th and j-th items. We denote this distance matrix by M and distance function between items by $d(i, j)$.

3. Let each item be in its own cluster, and we have N clusters each of which contains only one item.

4. Search the closest cluster pair from N clusters, then merge the pair into one cluster. Now we have $N - 1$ clusters. Let us denote distance function between two clusters c_k and c_l by $D(c_k, c_l)$.

5. Search the closest cluster pair from $N - 1$ clusters and merge them so that we have $N - 2$ clusters.

6. Repeat this "search and merge" operation until clusters are merged to single cluster which contains all N items.

7. Now, we have a binary tree of merging operation with $N - 1$ hierarchy. The root of the tree corresponds to the final single cluster with N items, and the leaves to the initial N clusters each of which contains only one item.

8. To find clusters located at least L distance apart, descend down from the root to the leaves until find a merging operation which merged two clusters in L or further distance.

Here, we use Euclidean distance as d and average-linkage as D since we simply cluster vectors representing physical positions and motions. In average-linkage clustering, D is given by:

$$D(c_k, c_l) = \frac{1}{N(c_k) \times N(c_l)} \left(\sum_{m \in c_k} \sum_{n \in c_l} d(m, n) \right), \tag{4.1}$$

where m and n denote a member of the cluster c_k and c_l respectively, and $N(c_k)$ and $N(c_l)$ denote the number of members in these clusters.

Using this hierarchical clustering algorithm, our motion clustering algorithm is defined as follows.

1. Suppose we have flow vectors as described in Section 3.4, each of which has its own position, normalized direction, and length (Figure 4.1(a)). We denote i-th flow vector by f_i, its position by $\boldsymbol{p}_i$, normalized direction by $\boldsymbol{d}_i$, and length by l_i.

2. First, filter out flow vectors such that length of which are under a certain threshold. In Figure 4.1(b), removed flows are illustrated as thin lines in light grey color and remaining flows as thick lines in black.

3. Next, apply hierarchical clustering based on the vector position $\boldsymbol{p}$. Figure 4.1(c) shows a clustering result and each of cluster is in different color.

4. Then, for each cluster, apply hierarchical clustering based on the vector normalized direction $\boldsymbol{d}$ (Figure 4.1(d)).

Note that we need three thresholds: one for filtering and two for clustering. These thresholds are determined based on the interval between frames and expected object motion.

Now, we obtained clusters each of which is estimated as rigid motion. We then extract rotation and translation parameters which describes a rigid motion. That is, for each cluster, we have the positions of each flow and their destinations, and we compute the rotation and translation between them.

Let us denote the positions by $P = \{\boldsymbol{p}_i\}$, the destinations by $Q = \{\boldsymbol{q}_i \mid \boldsymbol{q}_i = \boldsymbol{p}_i + l_i \boldsymbol{d}_i\}$. Rotation matrix $\mathbb{R}$ and translation vector $\boldsymbol{T}$ should minimize following error function:

$$\text{err} = \sum_{i=1}^{N} \|\boldsymbol{q}_i - (\mathbb{R}\boldsymbol{p}_i + \boldsymbol{T})\|^2, \tag{4.2}$$

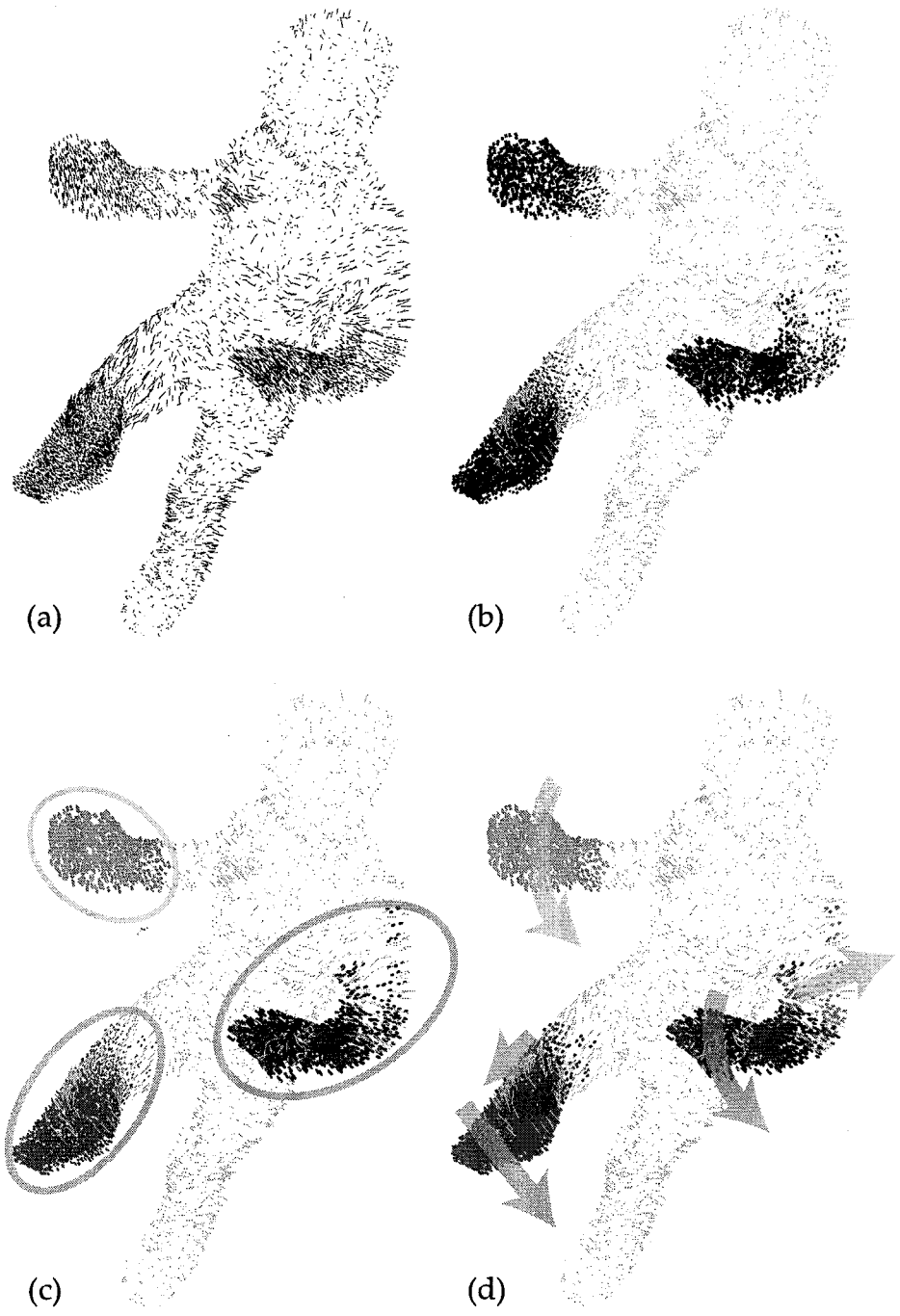


Figure 4.1: Clustering of the estimated motion flow vectors

where N is a number of flow vectors. First, we subtract the mean vector from each point set:

$$\hat{P} = \{\hat{p}_i \mid \hat{p}_i = p_i - \bar{p}\}, \tag{4.3}$$

where $\bar{p}$ is the centroid of P given by

$$\bar{p} = \frac{1}{N} \sum_{p_i \in P} p_i. \tag{4.4}$$

Similarly,

$$\hat{Q} = \{\hat{q}_i \mid \hat{q}_i = q_i - \bar{q}\}, \tag{4.5}$$

where

$$\bar{q} = \frac{1}{N} \sum_{q_i \in Q} q_i. \tag{4.6}$$

Rotation $\mathbb{R}$ which minimize Equation (4.2) is given as the largest eigenvalue λ of the following matrix in quaternion form [Hor87][WI95][OK98]

$$\begin{pmatrix} S_{xx}+S_{yy}+S_{zz} & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & S_{xx}-S_{yy}-S_{zz} & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & -S_{xx}+S_{yy}-S_{zz} & S_{yz}+S_{zy} \\ S_{xy}-S_{yx} & S_{zx}+S_{xz} & S_{yz}+S_{zy} & -S_{xx}-S_{yy}+S_{zz} \end{pmatrix}, \tag{4.7}$$

where

$$S_{xx} = \sum_{i=1}^{N} \hat{p}_{i,x} \hat{q}_{i,x}\ , \tag{4.8}$$

$$S_{xy} = \sum_{i=1}^{N} \hat{p}_{i,x} \hat{q}_{i,y}\ , \tag{4.9}$$

and so on. Note that $\hat{p}_{i,x}$ and $\hat{q}_{i,y}$ denote x and y element of the vector $\hat{p}_i$ and $\hat{q}_i$ respectively. Conversion from quaternion $\lambda = w + ix + jy + kz$ to matrix $\mathbb{R}$ is given by

$$\mathbb{R} = \begin{pmatrix} w^2+x^2-y^2-z^2 & 2(xy-zw) & 2(xz+yw) \\ 2(xy+zw) & w^2-x^2+y^2-z^2 & 2(yz-xw) \\ 2(xz-yw) & 2(yz+xw) & w^2-x^2-y^2+z^2 \end{pmatrix}, \tag{4.10}$$

and applying $\|\lambda\|^2 = w^2 + x^2 + y^2 + z^2 = 1$, we have

$$\mathbb{R} = \begin{pmatrix} 1-2(y^2+z^2) & 2(xy-zw) & 2(xz+yw) \\ 2(xy+zw) & 1-2(x^2+z^2) & 2(yz-xw) \\ 2(xz-yw) & 2(yz+xw) & 1-2(x^2+y^2) \end{pmatrix}. \tag{4.11}$$

Translation vector $\boldsymbol{T}$ is simply given by

$$\boldsymbol{T} = \bar{\boldsymbol{q}} - \mathbb{R}\bar{\boldsymbol{p}}. \tag{4.12}$$

Using this clustering and rotation / translation estimation, we can now categorize the vertices into two types:

Ca-1 vertices in warping, and

Ca-2 vertices in rigid motion.

For a vertex categorized as **Ca-2**, we have its rotation matrix and translation vector, so we replace its motion flow by these parameters:

Step 1. Suppose we have the rotation matrix $\mathbb{R}$, the translation vector $\boldsymbol{T}$, and the position of the original motion flow $\boldsymbol{p}$ for each vertex categorized as **Ca-2**.

Step 2. The destination $\boldsymbol{d}$ of new motion vector is given by

$$\boldsymbol{d} = \mathbb{R}\boldsymbol{p} + \boldsymbol{T}. \tag{4.13}$$

Step 3. Using spherical linear interpolation (or SLerp) for rotation and linear interpolation (or Lerp) for translation, we define new motion flow with parameter t:

$$\frac{\sin(\theta(1-t))}{\sin\theta}\boldsymbol{p} + \frac{\sin\theta t}{\sin\theta}\boldsymbol{d} + t\boldsymbol{T}, \tag{4.14}$$

where θ denotes the angle between $\boldsymbol{p}$ and $\boldsymbol{d}$.

4.4 Vertex Categorization Based on its Identifiability

As is well known, we can not expect that all the points on the object surface have prominent texture, nor can recover all their 3D position by stereo method. Hence not all the vertices of the mesh model are *identifiable* or *localizable*. So the photometric force $\boldsymbol{F}_e(v)$ (Section 3.5.1), which put a vertex on the real object surface

based on texture correlation, will not work at such vertices. Here we assume that we can categorize the vertices into two types based on their surface properties:

- a vertex with prominent texture, or
- otherwise.

As described in previous chapters, the photometric force $\boldsymbol{F}_e(v)$ is the component which explicitly estimate the object shape. This categorization states that we can classify the vertices into following two types:

Cb-1 vertices which should lead themselves and their neighbors so as to be placed so that they have consistent textures, and

Cb-2 vertices which should be led by others.

We regard a vertex as identifiable if it has consistent and prominent textures in visible cameras. Here, we introduce an identifiability-scoring function $I(v)$ as follows:

$$I(v) \equiv E_e(\boldsymbol{q}_v) \times \min \left\{ \min_{c \in C_{\hat{v}}} \nabla p_{\hat{v},c},\ \min_{c \in C_v} \nabla p_{v,c} \right\}, \tag{4.15}$$

where $E_e(\boldsymbol{q}_v)$ denotes the correlation of textures of v (see Equation (3.9)), $\nabla p_{\hat{v},c}$ and $\nabla p_{v,c}$ the derivatives of the texture of $\hat{v}$ and v on camera c respectively. With this function $I(v)$, we compute the identifiability for each vertex, and label as **Cb-1** (identifiable) if the score exceeds a certain threshold, and as **Cb-2** if not.

4.5 Heterogeneous Deformation Based on Vertex Categorization

We introduced two types of vertex categorization. One is based on the motion clustering from physical point of view, and the other one is based on the surface property. With these two categorizations, we add following steps in the basic inter-frame deformation process in Chapter 3.

For each vertex,

- if is it categorized as **Cb-1**, let the force of the vertex diffuse to those of neighbors so that it lead its neighbors,
- and / or if categorized as **Ca-1**, make the springs of the vertex stiff[Pro95][CMN97] to move together with others.

We define the diffusion process as follows.

1. Let v be a vertex of type **Cb-1**, v_j a neighboring vertex of v.

2. For each v_j, modify the force $F(v_j)$:

$$\begin{aligned} \boldsymbol{F}(v_j) &= \omega \boldsymbol{F}(v) + (1-\omega)\boldsymbol{F}_{\text{prev}}(v_j), \\ \omega &= e^{-D_g(v,v_j)}, \end{aligned} \tag{4.16}$$

   where $\boldsymbol{F}_{\text{prev}}(v_j)$ denotes the original force at v_j, and $D_g(v, v_j)$ the geodesic distance between v and v_j.

Note that for a vertex of type **Ca-2** $\wedge$ **Cb-2**, a vertex without prominent texture or not a part of a rigid part, its position is interpolated by the internal force $\boldsymbol{F}_i(v)$, and a vertex of type **Ca-1** $\wedge$ **Cb-1**, a vertex with prominent texture and a part of a rigid part, deforms so as to lead the rigid part which the vertex belongs to.

Applying these steps, our heterogeneous deformation process is modified as follows:

Step 1. Suppose we have an initial mesh model representing the object shape at frame t. If we have been in successive inter-frame deformation, use the mesh deformed from previous frame $t-1$, otherwise, use the intra-frame deformation (in Chapter 2) to obtain the initial shape.

Step 2. Compute the initial likelihood of contour generator $\hat{L}(v)$ for each vertex (in Section 3.5.2).

Step 3. Compute roughly estimated motion flow for the drift force $\boldsymbol{F}_d(v)$ and the inertia force $\boldsymbol{F}_n(v)$.

Step 4. Estimate the motion flow vector for $\boldsymbol{F}_d(v)$ (in Section 3.5.4).

Step 5. Categorize the vertices based on the motion flow:

  Step 5.1. By clustering the estimated motion flow, label the vertex whether **Ca-1**: it is an element of a rigid part, or **Ca-2**: it is not.

  Step 5.2. Make the springs of vertices labeled as **Ca-1** stiff.

Step 6. Deform the model iteratively:

  Step 6.1. Compute forces working at each vertex respectively.

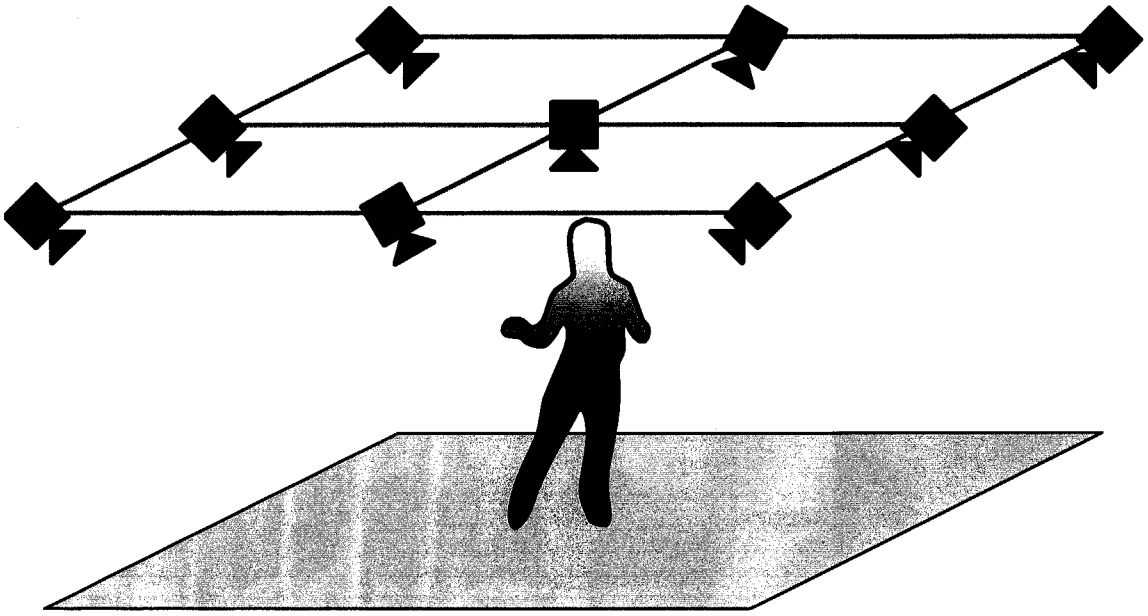


Figure 4.2: Camera arrangement

Step 6.2. For a vertex whose identifiability $I(v)$ exceeds a certain threshold, that is, for a vertex labeled as **Cb-1**, let the force of it diffuse to those of neighbors.

Step 6.3. Move each vertex according to the force.

Step 6.4. Terminate if the vertex motions are small enough. Otherwise go back to Step 6.1.

Step 7. Take the final shape of the mesh model as the object shape at frame $t + 1$.

4.6 Experimental Results

Figure 4.3, 4.4, and 4.5 illustrate the inter-frame deformation through eight successive frames. The columns of Figure 4.3 show, from left to right, the captured images, the visual hulls generated by the frame-wise discrete marching cubes method, and the mesh models deformed by the heterogeneous deformation method, respectively. In these two figures, colored areas of the mesh denote rigid parts of the object estimated by the clustering at **Step 3**. Note that the visual hull in frame t was used as the initial shape for our deformation. In this experiment, we used 9 cameras circumnavigating the object (Figure 4.2). Captured multi-viewpoint videos are not completely synchronized and include mo-

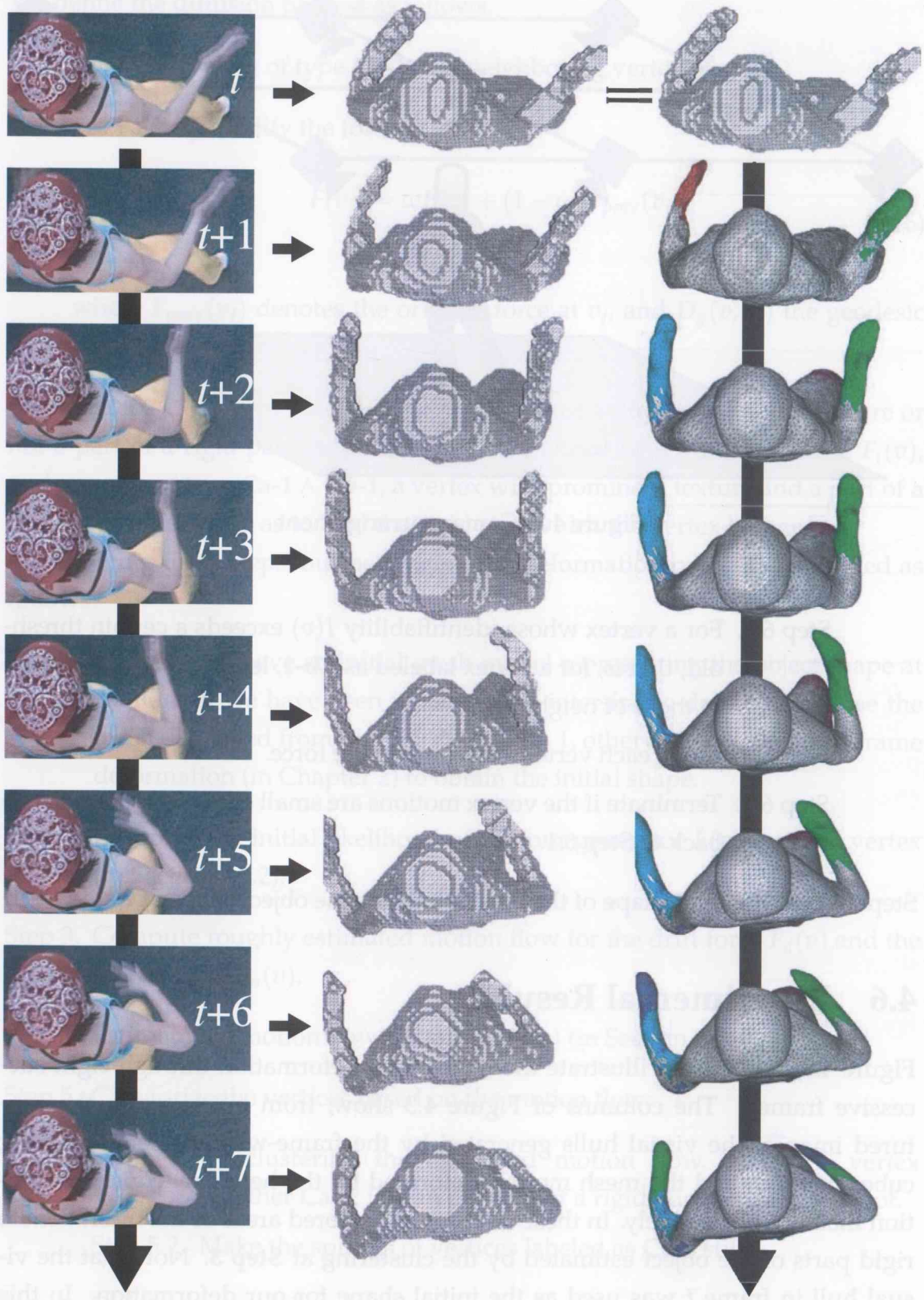


Figure 4.3: Successive deformation results

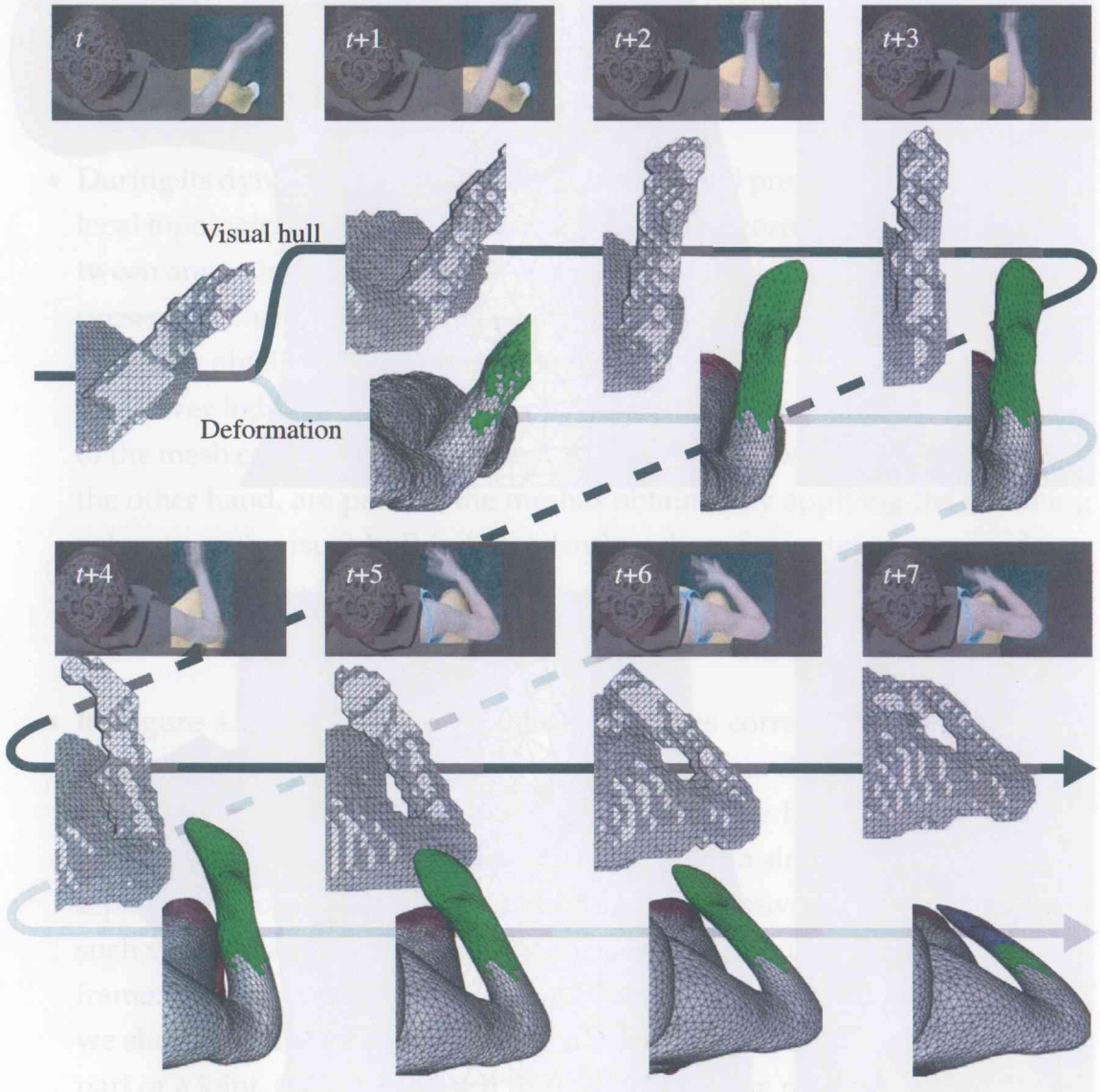


Figure 4.4: Successive deformation results (detailed)

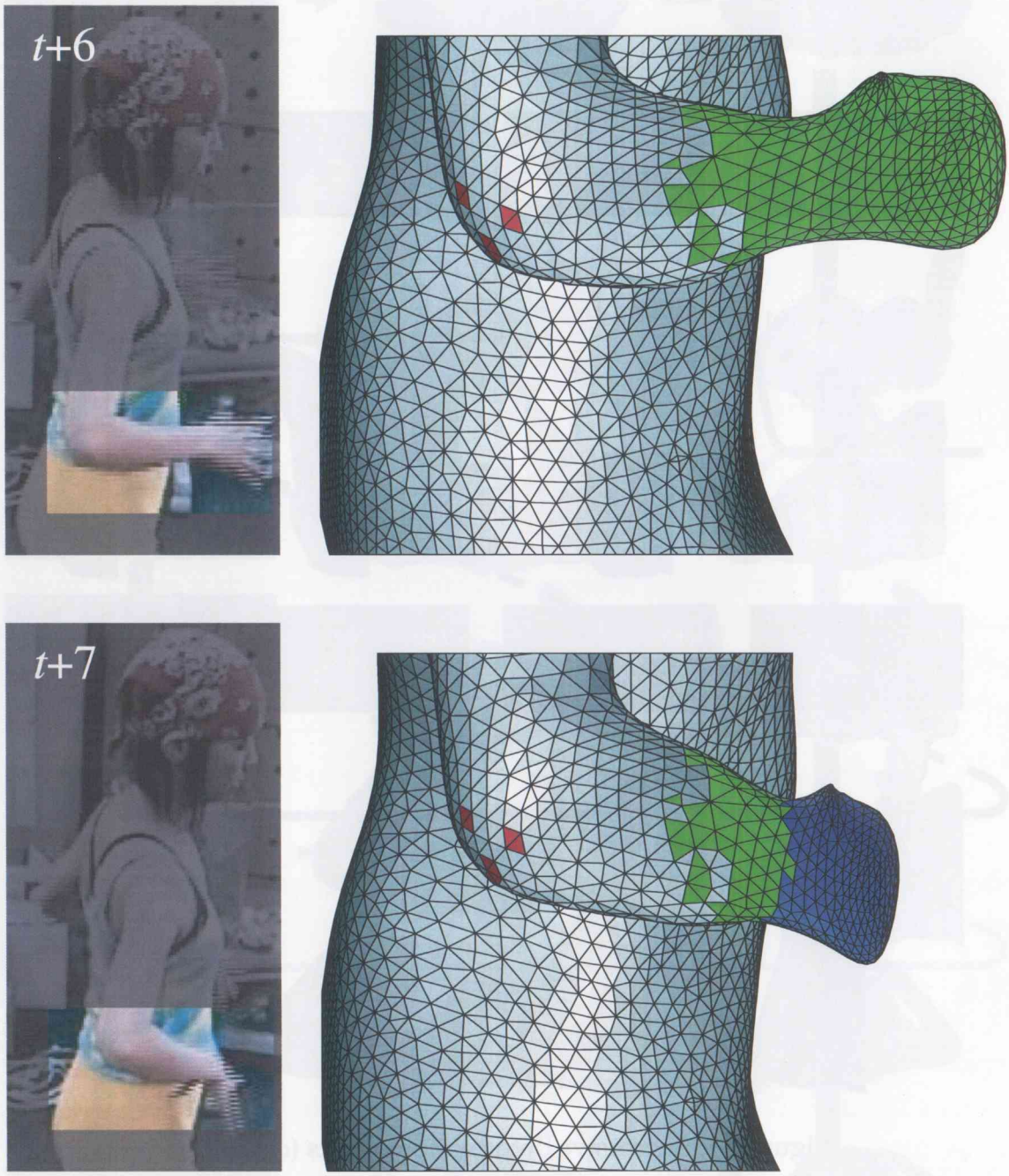


Figure 4.5: Successive deformation results (detailed, side view)

tion blur. The mesh models consist of about 12,000 vertices and 24,000 triangles, and the processing time per frame is about 20 minutes by PC (Xeon 3.0GHz). We used fixed coefficients $\alpha = 0.2, \beta = 0.2, \gamma = 0.2, \delta = 0.3, \epsilon = 0.1$ given a priori.

From these results, we can observe:

- Our deformable mesh model can follow the partially-rigid object motion smoothly. In Figure 4.3 and 4.4, we can observe that the arms of the object are labeled as rigid regions.

- During its dynamic deformation, our mesh model preserves both global and local topological structure and hence we can find corresponding vertices between any pair of frames for all vertices. Figure 4.4 illustrates this topology preserving characteristic. That is, the top-left mesh denotes a part of the initial mesh obtained by applying the marching cubes to the visual hull at t. The lower light arrow stands for our deformation process, where any parts of the mesh can be traced over time. Aligned along the upper dark arrow, on the other hand, are parts of the meshes obtained by applying the marching cubes to each visual hull independently, where no vertex correspondence can be established because the topological structures of the mesh data are different.

- In Figure 4.5, we can observe that the vertices corresponding to the right lower arm and hand of the object were labeled into different groups (green and blue) at frame $t + 7$ because the arm and the hand moved to different directions. This tells that vertices labeled as a single rigid part may be separated into two or more parts through successive frame recovery, and such vertices labeled into different parts may be grouped together in future frame. That is, we can find new joint of the object through the recovery, and we should introduce a deformation model which can learn where is a rigid part or a joint, and can utilize it to accomplish more robust reconstruction.

4.7 Summary

In this chapter, we proposed a computational framework using a heterogeneous deformable mesh model based on vertex categorization. We categorized vertices based on their physical and photometric properties.

- By clustering the estimated motion flow, we categorized vertices into rigid-motion part and warping part.
- Using texture-prominency of each vertex, we categorized vertices whether identifiable (or localizable) or not.

According to these two categorization, we changed the deformation process as follows:

- For each vertex such that it is a part of rigid-motion, make its spring-constant stiff to deform together with its neighbors.
- For each vertex such that it is labeled as identifiable, let its force diffuse to its neighbors so that it can lead its neighbors.

The basic inter-frame deformation in Chapter 3 was totally per-vertex and uniform deformation, but heterogeneous deformation is an introduction of group-wise approach to make the deformation stable.

Chapter 5

Dynamic 3D Shape Estimation for Object with Time-Varying Global Mesh Topology

In this chapter, we introduce an algorithm to estimate the object shape and motion between two frames with different global mesh topology. That is, we propose a method to deform a mesh model representing the object shape in genus-0 (Figure 3.1, top row) to the genus-1 or 2 shape (Figure 3.1, middle and bottom rows).

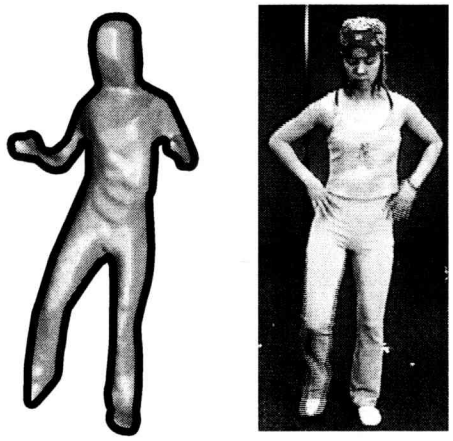
5.1 Problem Description

The problem we consider is 3D shape and motion estimation of the object between two successive frames with time-varying global topology. We assume the object which has

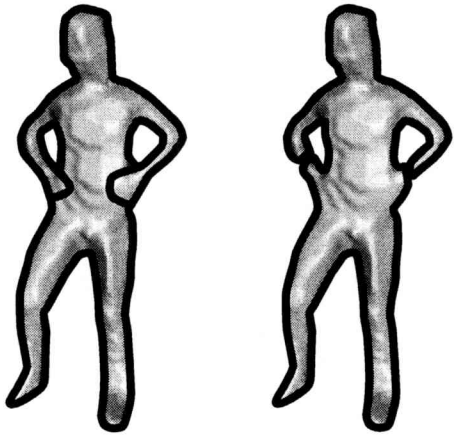
- arbitrary but smooth and continuous shape,
- arbitrary motion, but can be categorized into warping and rigid motion,
- Lambertian surface reflectance, and
- time-varying global topology.

Here time-varying global topology means that the genus of the object in two frames can change.

We represent the object shape by deformable mesh model, and the object motion by translations of vertices composing the mesh. The input data of our algorithm are:



(a) mesh at frame t (b) object image at frame t+1



(c) defomed mesh (d) defomed mesh

Figure 5.1: Deformation for object with variable global topology

- multi-viewpoint images and object silhouettes at both frame t and $t+1$, and
- a mesh model representing the object shape at frame t.

The output data are:

- the object shape at frame $t+1$ represented by a mesh model, and
- the object motion from t to $t+1$ represented as translations of vertices.

5.2 Object with Time-Varying Global Topology

In deforming a mesh model so as to represent the object shape in another global topology, we have to consider not only the geodesic proximity but also the Eu-

clidean proximity between vertices. Suppose we have a mesh in Figure 5.1(a) as the object shape at frame t, and deform it so as to be the object shape at frame $t+1$ (Figure 5.1(b)). We can observe that the mesh topology will change from t to $t+1$ and a hole will appear at the "waist". In this situation, many papers on active contour model tried to develop algorithms to change the global topology of the mesh model to be equal to that of the object[DM01][BLS03][YS03b], but it is advisable for our objective to keep the global topology of the mesh model and hidden surfaces. That is, we deform our mesh model so that the "hand" of the mesh deform to touch its body, since it is the actual object motion (Figure 5.1(c)). However, deformation algorithm in Chapter 4 may produce a different result as shown in Figure 5.1(d), i.e., the "hand" will be smaller and a protuberance sticks out from the "waist".

This is because that we have assumed that there are a positive correlation between geodesic distance and Euclidean distance for each vertex. Here, geodesic distance between two vertices is the length of the shortest path on the mesh surface between them. Based on this assumption, we defined the vertex forces so that they care the vertices only in their geodesic vicinity. However, if the global mesh topology changes as described above, this assumption will break down and there will be a negative correlation between geodesic and Euclidean distances.

Since the heterogeneous deformation algorithm in the previous chapter is defined implicitly based on this proximity assumption,

- the motion estimation algorithm in Section 3.4.1 may generate inappropriate displacement vectors for visual hulls with different global topology, and
- silhouette constraint cannot resolve an ambiguity about contour generators.

This is because they blindly search nearest point as its destination for all vertices based on proximity assumption, and if the global topologies of two visual hulls are different, it takes inappropriate destinations for regions to be touched. Figure 5.2 illustrates this situation. Suppose we have two visual hulls having different global topology (Figure 5.2(a) and (b), red and blue points). For points in touching / occluding region (in green), displacement vectors (red arrows) would be inappropriate as illustrated in 5.2(c) since such occluded region cannot be observed by cameras at frame $t+1$, and cannot be represented by visual hull.

Next, we discuss about an ambiguity about contour generators. Silhouette preserving force $F_s(v)$ (Section 3.5.2) which estimates contour generators on the object surface may also behave similarly to what described above. Figure 5.3

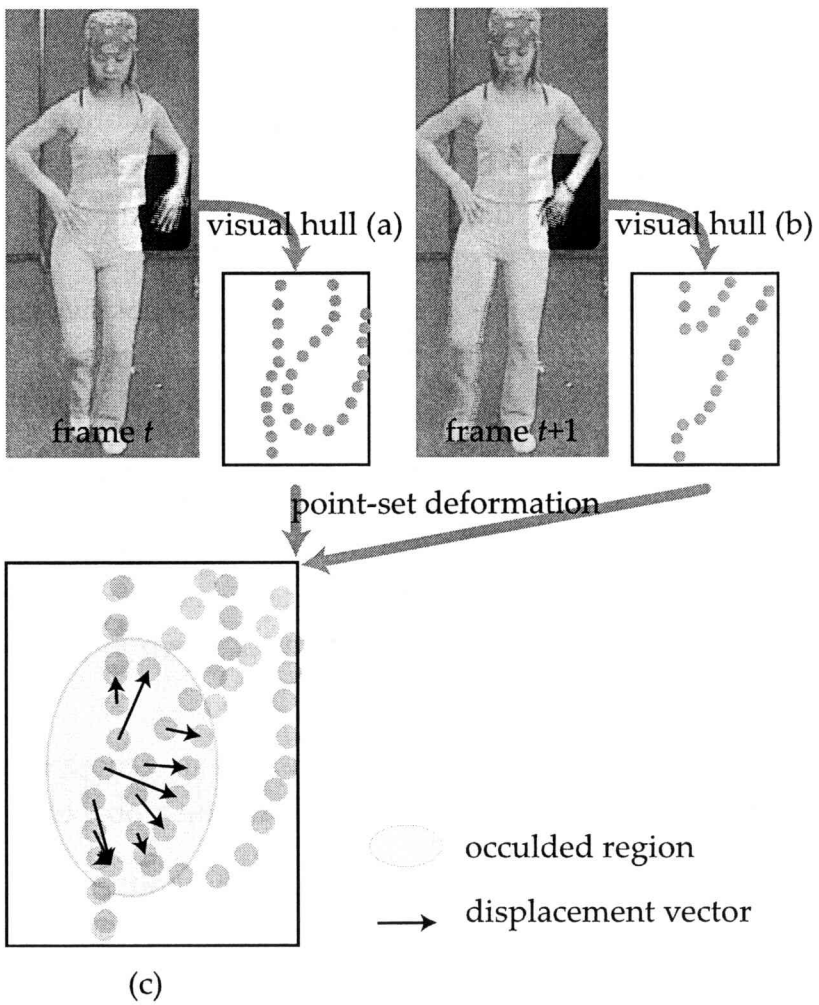


Figure 5.2: Naive point-set deformation algorithm applied to visual hulls with different global mesh topologies

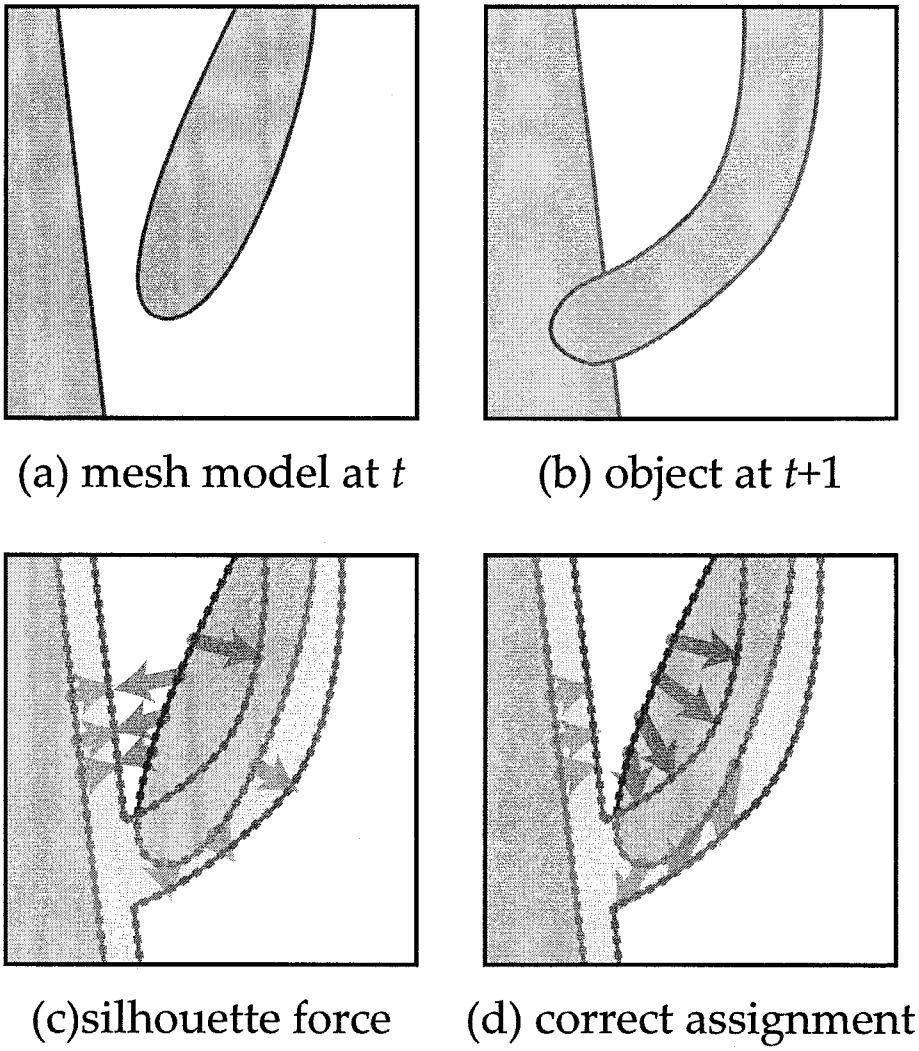


Figure 5.3: Silhouette preserving force in touching

explains this situation. Suppose we have a mesh model representing the object shape at frame t (Figure 5.3(a)), and now we are about to deform it toward frame $t+1$ (Figure 5.3(b)). In this situation, the silhouette preserving force $F_s(v)$ first assigns two mesh contours to one observed silhouette contour (Figure 5.3(c)) while only one mesh contour should be assigned as shown in Figure 5.3(d) at the end of deformation process. That is, both two candidates will deform to be a contour generator, and silhouette preserving force $F_s(v)$ cannot select correct one from these candidates at the begging of the deformation. Figure 5.4 shows another situation but explained as same as above. Suppose we have a mesh model representing the object shape at frame t (Figure 5.4(a)), and now we are about to deform it toward frame $t+1$ (Figure 5.4(b)). In this situation, the object will not touch itself, but from a certain viewpoint, its "hand" self-occludes its "body" (Figure 5.4(c)). Similarly to what shown in Figure 5.3, silhouette preserving force $F_s(v)$ takes two mesh contours as candidates to one apparent contour (Figure 5.4(d)).

5.2.1 Solution

To solve this problem, we propose following approach.

1. First of all, we consider that we cannot distinguish / predict correct contour generators from two or more candidates at the begging of the deformation.

2. Obviously, we need new vertex force to avoid collisions. Let us introduce a new vertex force working at such a collision part so that two collided surfaces push each other. We call this new force as repulsive force.

3. Suppose two surfaces which are contour generator candidates collide as illustrated in Figure 5.5(c).

4. Recall that the silhouette preserving force keeps the mesh to be encaged by the visual hull, and the internal force keeps the mesh shape be smoothly interpolated.

5. So if we leave occluded surface to deform by the internal force, the silhouette preserving force and the repulsive force (Figure 5.5(c), red and green arrows) will push back such a collided surfaces to be inside of the visual hull (Figure 5.5(d)).

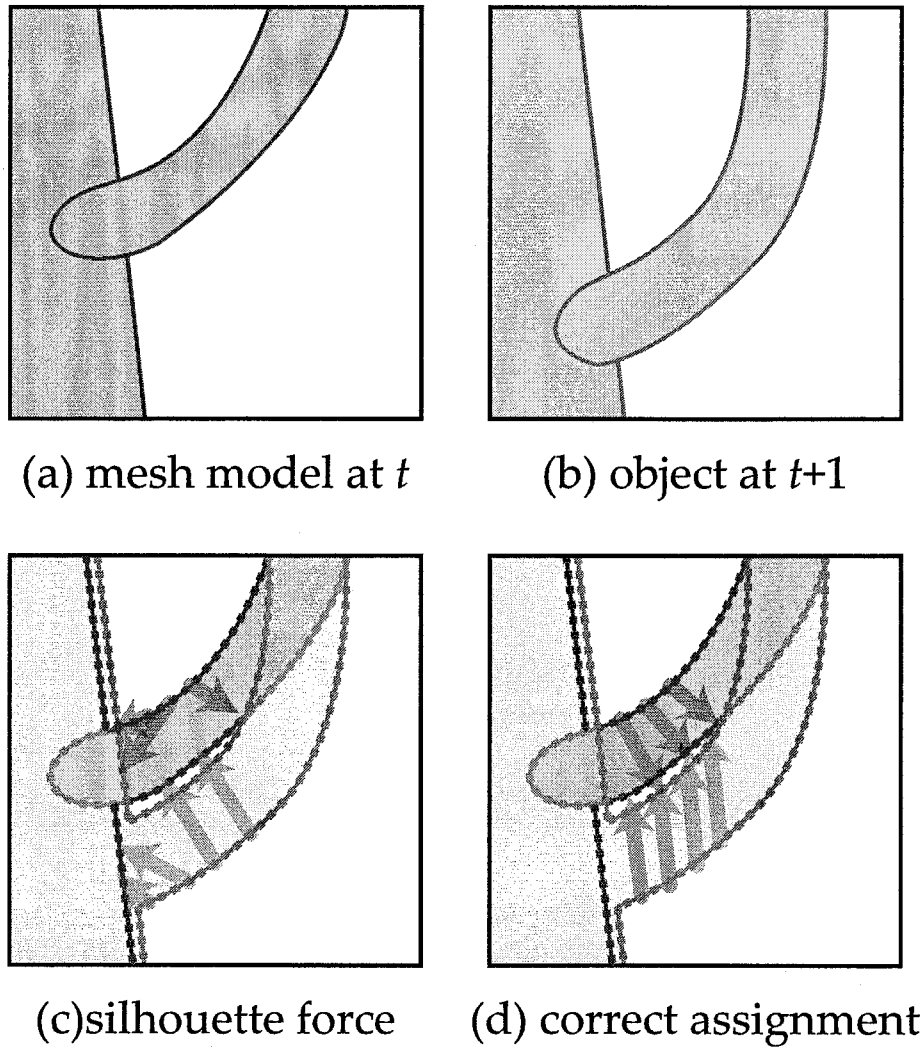


Figure 5.4: Silhouette preserving force in occluding

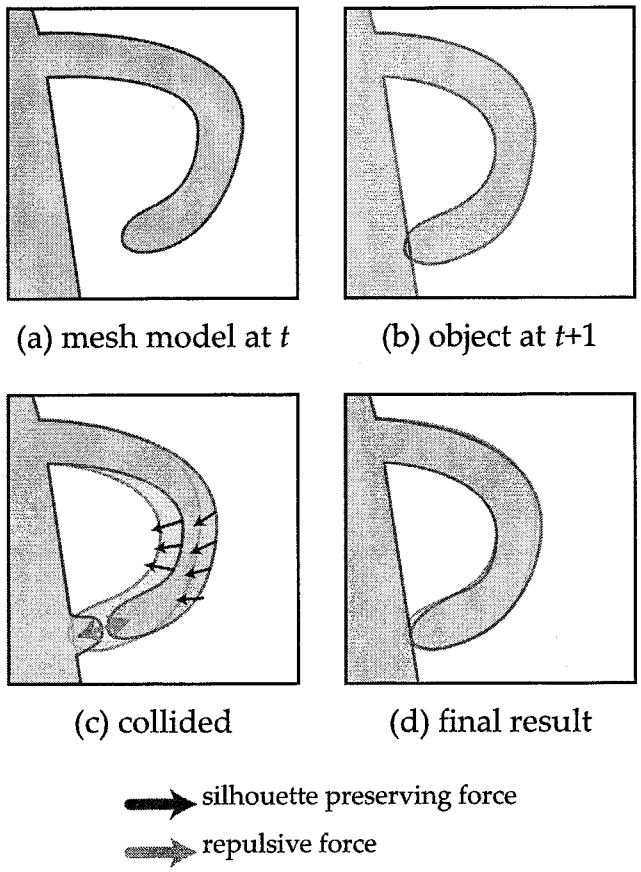


Figure 5.5: Repulsive force at collision

6. That is, even if we have multiple candidates, silhouette force makes the surface which is outside of the observed silhouette to be the final contour generator and repulsive and internal force push back the other part.

Following sections describes how we modify the motion estimation algorithm in Section 3.4.1, and the definition of repulsive force.

5.3 Motion Estimation from Two Visual Hulls with Different Global Mesh Topology

As described in Section 5.2 and shown in Figure 5.2, naive approach in Section 3.4.1 will produce inappropriate motion flows when two visual hulls have different global mesh topology. This is because that *touched* surface cannot be represented by the visual hull, and voxels of *in-touching* surface cannot find their correspondings in it.

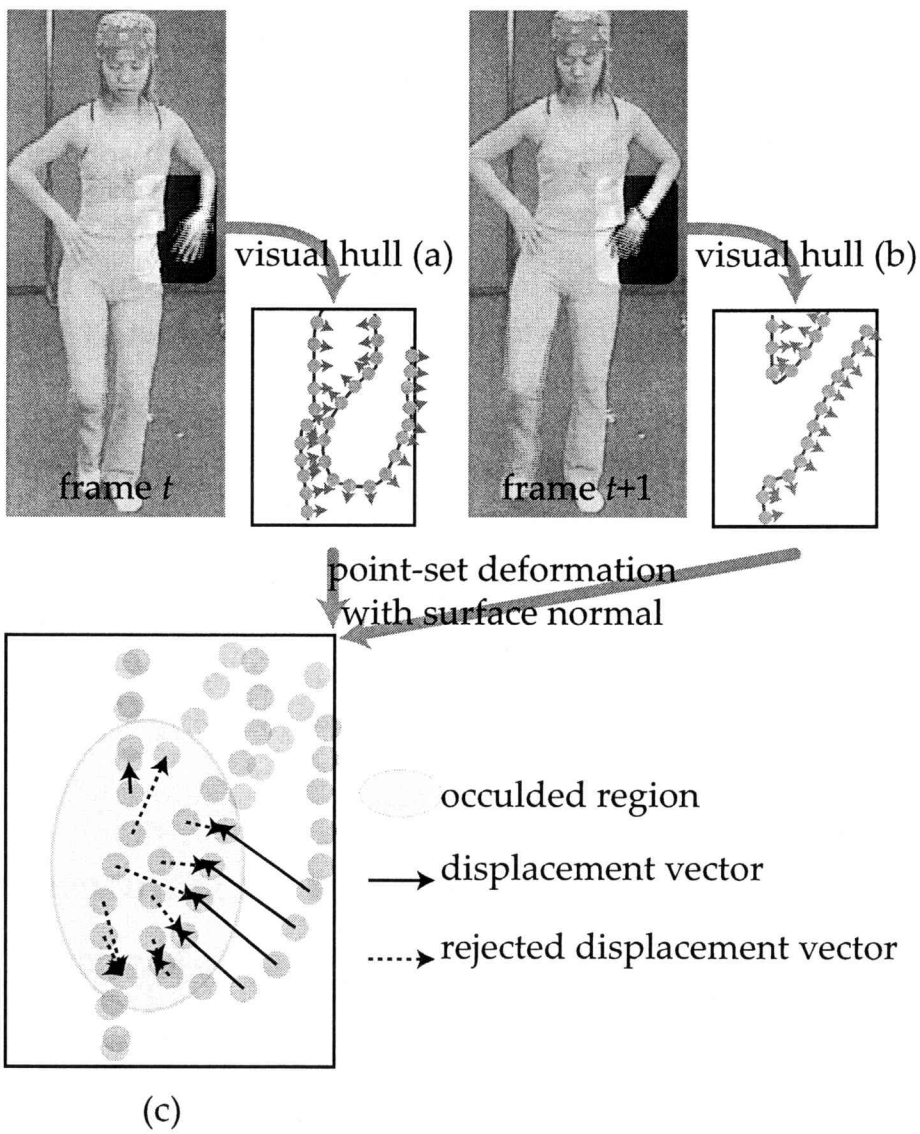


Figure 5.6: Motion estimation with surface normal consideration

The basic strategy to solve this problem is culling out of such miss-correspondings by considering the surface normal consistency. Suppose we have visual hulls in two frames (Figure 5.6(a) and (b)). In Section 3.4.1, we simply used these visual hulls as sets of voxels, however, we use them as surface models which have vertices and normals to enable the culling. Our culling algorithm using surface normals is defined as follows:

1. Compute $D(\boldsymbol{a}_i, B)$ and $D(\boldsymbol{b}_j, A)$ (Equation (3.3) and (3.4) respectively) for each vertex of visual hulls as usual (Figure 5.6(c), both solid and dotted arrows in orange).

2. For each point, compute the inner product of normal vectors between corresponding points, and if it is under a certain threshold, let $D(\cdot) = 0$ (Figure 5.6(c), dotted arrows in orange).

So we redefine Equation (3.3) and (3.4) as:

$$D(\boldsymbol{a}_i, B) = \frac{1}{1 + \exp\left(-\sigma \boldsymbol{n}_{\boldsymbol{b}_{j_{a_i}}} \cdot \boldsymbol{n}_{\boldsymbol{a}_i}\right)} \left(\boldsymbol{b}_{j_{a_i}} - \boldsymbol{a}_i\right), \tag{5.1}$$

and

$$D(\boldsymbol{b}_j, A) = \frac{1}{1 + \exp\left(-\sigma \boldsymbol{n}_{\boldsymbol{b}_j} \cdot \boldsymbol{n}_{\boldsymbol{a}_i}\right)} \left(\boldsymbol{b}_j - \boldsymbol{a}_i\right), \tag{5.2}$$

with sigmoid function $\frac{1}{1+\exp(-\sigma x)}$.

5.4 Collision Detection and Repulsive Force

As described in Section 5.2, we introduce new vertex force $F_r(v)$ which represent a repulsive force between surfaces in touching.

Collision detection is a well-known problem in cloth simulation of computer graphics or another physics based simulation. There are several "short-cuts" to detect a collision between special elements, e.g., spheres or functional surfaces, but collision detection for generic triangle meshes falls back basically to a kind of brute-force algorithm.

In this section, however, we propose a short-cut of collision detection for our deformable mesh model. First of all, since we have already introduced the internal force $F_i(v)$ to prevent a local collision, we focus only on a global collision

detection, which is a collision between surfaces geometrically close but topologically apart. Let us recall that in our deformation, global collisions occurs only at surfaces in touching. On such a touching surface, we can assume that visible cameras C_v for each vertex to be an empty set $\emptyset$. So we can drastically cull out vertices such that $C_v \neq \emptyset$ from collision detection target. After this effective culling, we apply brute-force algorithm.

We define our collision detection and repulsive force generation algorithm as follows:

Step 1. For all vertices in the mesh, initialize the repulsive force $\boldsymbol{F}_r(v)$ to 0.

Step 2. Suppose we have a set of vertices such that $V_\emptyset = \{v \mid C_v = \emptyset\}$.

Step 3. For each vertex $v \in V_\emptyset$,

Step 3.1. Compute the Euclidean distances to all the others.

Step 3.2. Find vertices such that they are within less than $l_{\min}(v)$ distance, where $l_{\min}(v)$ denotes the minimal length of edges connecting to v. Let $V_d(v)$ denote the set of vertices found for v.

Step 3.3. For each vertex $v' \in V_d(v)$, add following partial repulsive force $f_r(v, v')$ to $\boldsymbol{F}_r(v)$:

$$f_r(v, v') = \frac{q_{v'} - q_v}{\|q_{v'} - q_v\|^3}. \tag{5.3}$$

5.5 Deformation Process

Finally, we have following vertex force with 6 coefficients:

$$\boldsymbol{F}(v) \equiv \alpha \boldsymbol{F}_i(v) + \beta \boldsymbol{F}_e(v) + \gamma \boldsymbol{F}_s(v) + \delta \boldsymbol{F}_d(v) + \epsilon \boldsymbol{F}_n(v) + \zeta \boldsymbol{F}_r(v). \tag{5.4}$$

Using this vertex force, we define our final deformation process as follows:

Step 1. Suppose we have an initial mesh model representing the object shape at frame t. If we have been in successive inter-frame deformation, use the mesh deformed from previous frame $t-1$, otherwise, use the intra-frame deformation (in Chapter 2) to obtain the initial shape.

Step 2. Compute the initial likelihood of contour generator $\hat{L}(v)$ for each vertex (in Section 3.5.2).

Step 3. Estimate the motion flow vector for $F_d(v)$ (in Section 3.5.4 and 5.3).

Step 4. Categorize the vertices based on the motion flow (in Section 4.3):

- Step 4.1. By clustering the estimated motion flow, label the vertex whether **Ca-1**: it is an element of a rigid part, or **Ca-2**: it is not.
- Step 4.2. Make the springs of vertices labeled as **Ca-1** stiff.

Step 5. Deform iteratively. For each iteration,

- Step 5.1. Compute forces working at each vertex respectively.
- Step 5.2. For a vertex detected as in collision, let its coefficient of the silhouette force be 0.
- Step 5.3. For a vertex whose identifiability $I(v)$ exceeds a certain threshold, that is, for a vertex labeled as **Cb-1**, let the force of it diffuse to those of neighbors.
- Step 5.4. Compute velocities of vertices according to Equation (3.22).
- Step 5.5. Update positions of vertices according to Equation (3.23).
- Step 5.6. Terminate if the vertex motions are small enough. Otherwise go back to 5.1 .

Step 6. Take the final shape of the mesh model as the object shape at frame $t + 1$.

5.6 Evaluation

5.6.1 Experimental Results with Synthesized Images

Figure 5.8 shows deformation results of synthesized object using 9 cameras arranges as shown in Figure 5.7. The left column shows synthesized objects for each frame, the center column shows visual hulls of the object, and the right column shows deformed mesh models. Figure 5.9 shows average shape error between synthesized object and (a) visual hull, (b) the results of intra-frame deformation, and (c) the results of inter-frame deformation. Here, average shape error is defined as the average distance from each vertex to the nearest point in the synthesized object. Note that we used fixed coefficients $\alpha = 0.15, \beta = 0.15, \gamma = 0.2, \delta = 0.2, \epsilon = 0.1, \zeta = 0.2$ given a priori.

From these results, we can observe that

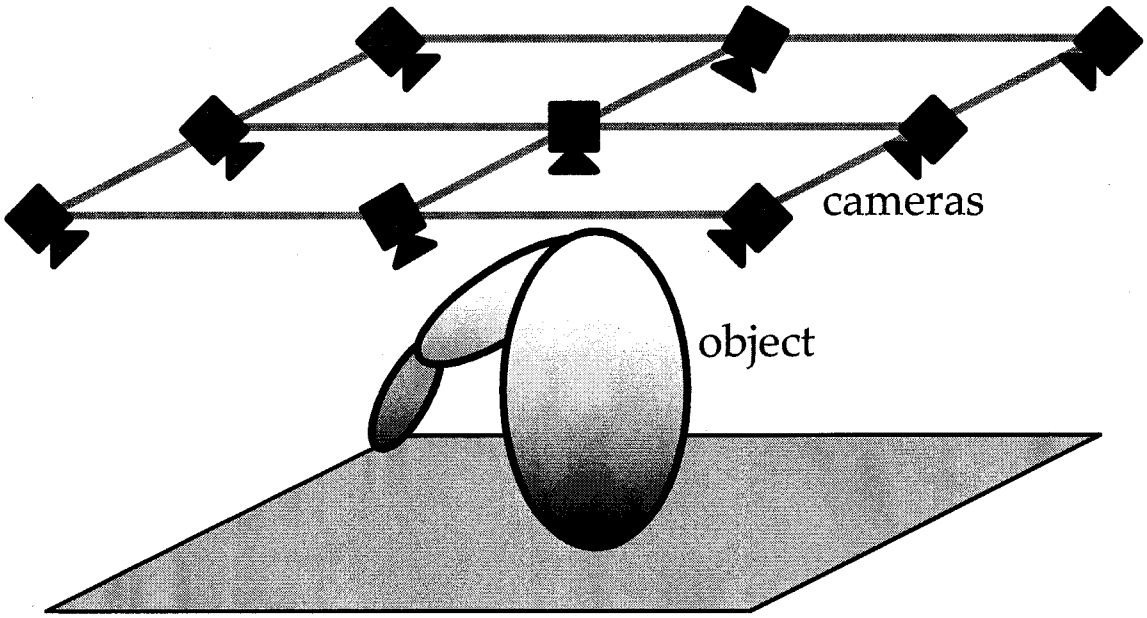


Figure 5.7: Camera arrangement

- The deformable mesh model can cope with global change of topology, i.e., can "touch" itself.
- At "elbow" of each deformation results, we can find folded surfaces in the inter-frame deformation results while the visual hulls and the intra-frame deformation results do not have such folds. This is because that such region is represented as a visible surface at the first frame, but it is turned to be invisible after some frames.
- The results of the intra-frame deformation are better than those of the inter-frame deformation (Figure 5.9). This is because that the inter-frame deformation cannot avoid error accumulation.

5.6.2 Experimental Results with Real Images

Figure 5.11 and 5.12 illustrate the inter-frame deformation for the object with time-varying global topology.

The columns of Figure 5.11 show, from left to right, the captured images, the visual hulls generated by the frame-wise discrete marching cubes method, the mesh models deformed by the intra-frame deformation method, and the mesh models deformed by the inter-frame deformation method respectively. In this

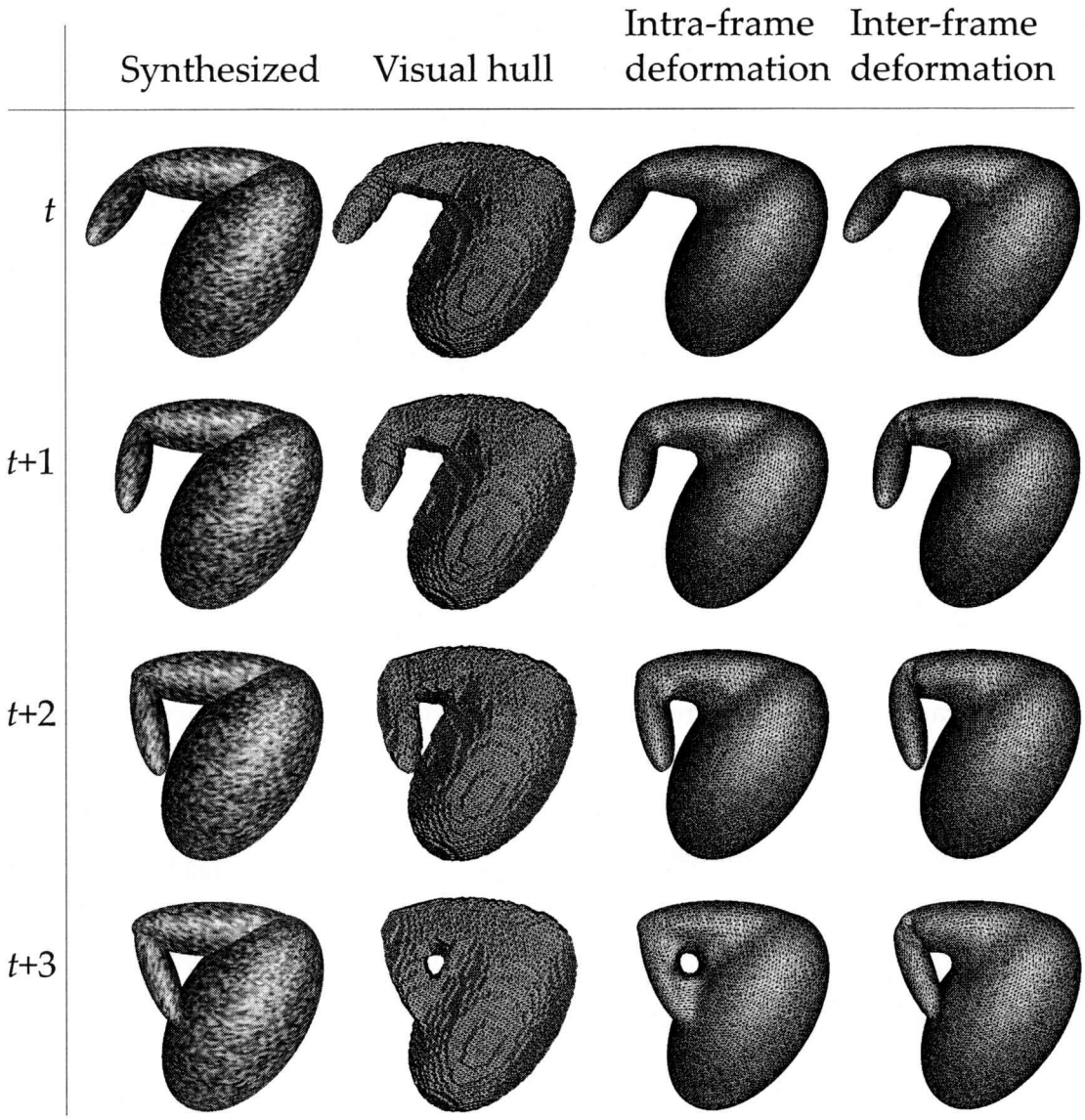


Figure 5.8: Estimation results of synthesized object

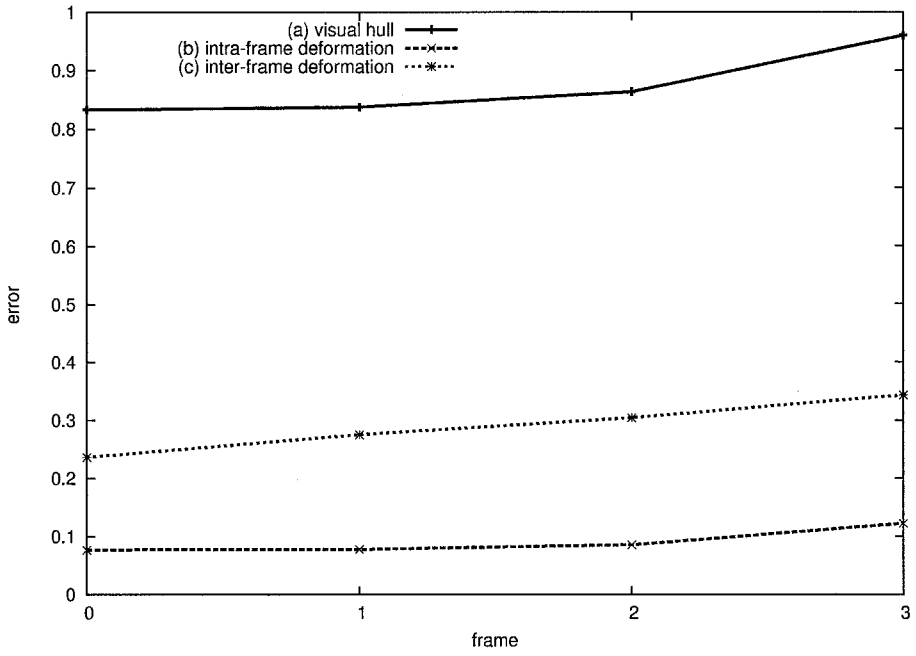


Figure 5.9: Average shape error

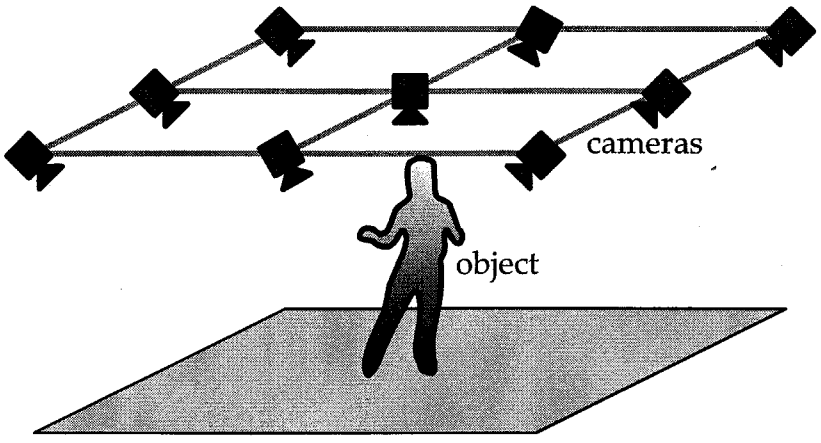


Figure 5.10: Camera arrangement

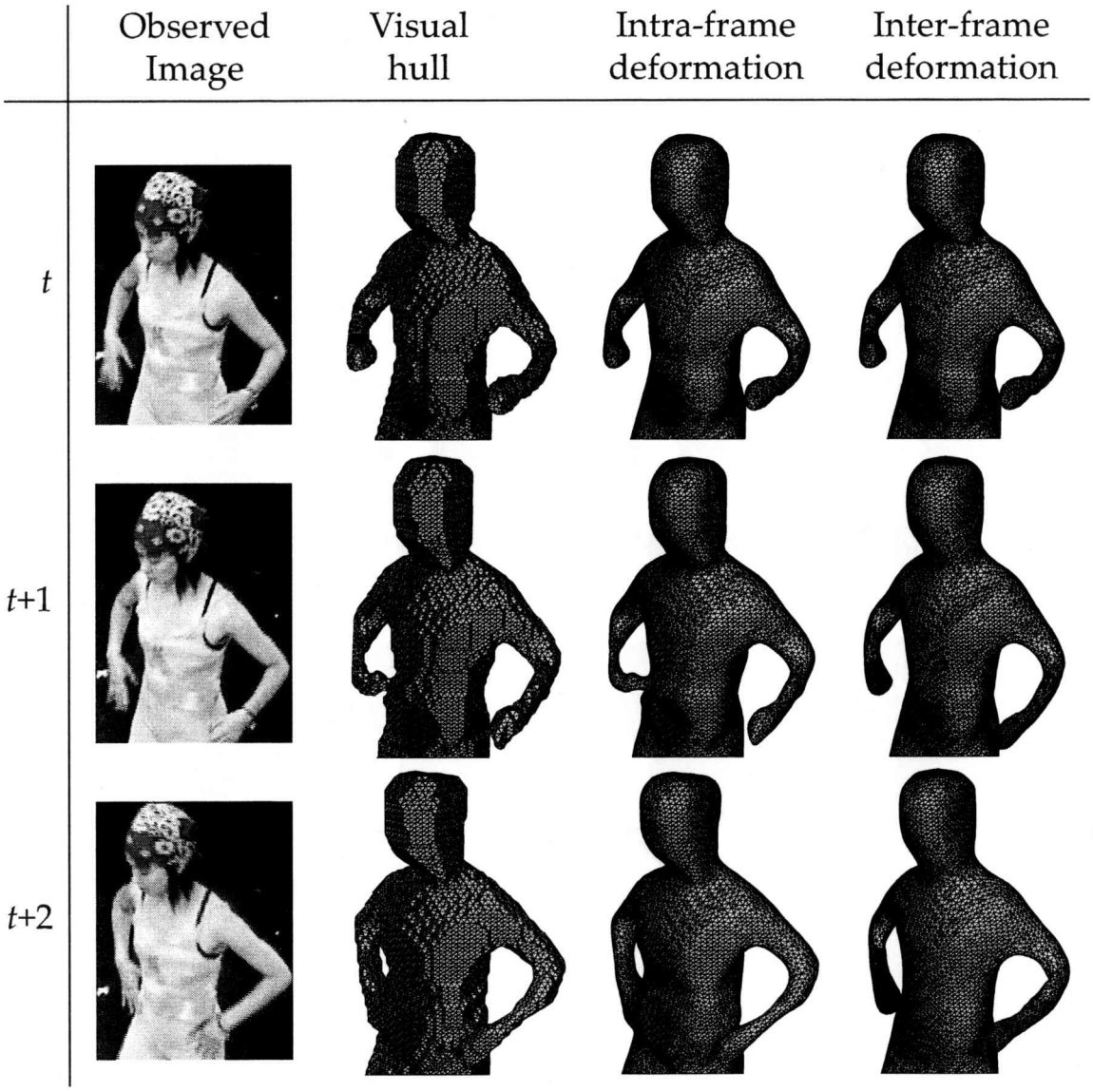


Figure 5.11: Successive deformation results

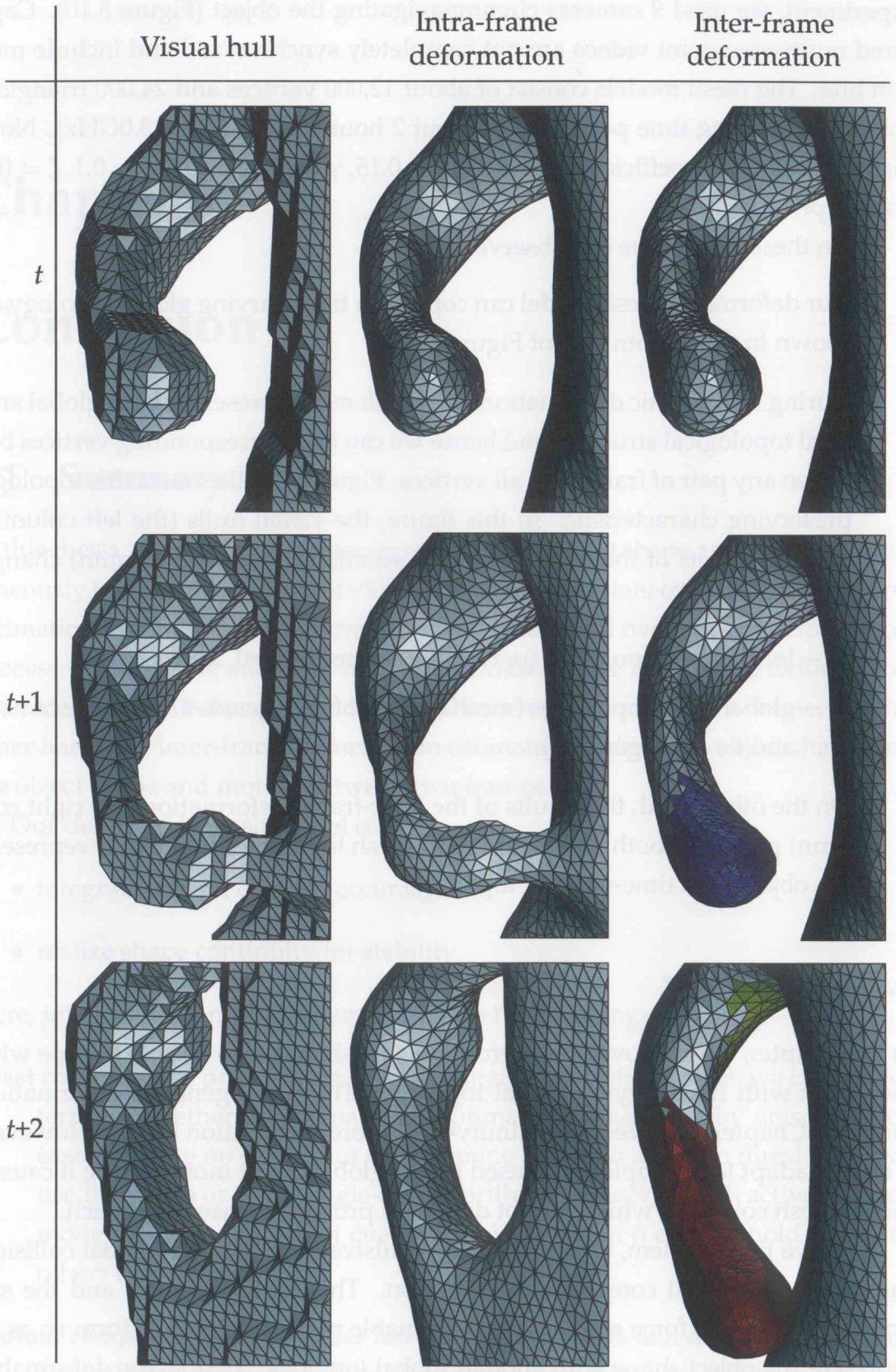


Figure 5.12: Successive deformation results (detailed)

experiment, we used 9 cameras circumnavigating the object (Figure 5.10). Captured multi-viewpoint videos are not completely synchronized and include motion blur. The mesh models consist of about 12,000 vertices and 24,000 triangles, and the processing time per frame is about 2 hours by PC (Xeon 3.0GHz). Note that we used fixed coefficients $\alpha = 0.15, \beta = 0.15, \gamma = 0.2, \delta = 0.2, \epsilon = 0.1, \zeta = 0.2$ given a priori.

From these results, we can observe:

- Our deformable mesh model can cope with time-varying global topology as shown in the bottom row of Figure 5.11.
- During its dynamic deformation, our mesh model preserves both global and local topological structure and hence we can find corresponding vertices between any pair of frames for all vertices. Figure 5.12 illustrates this topology preserving characteristic. In this figure, the visual hulls (the left column) and the results of the intra-frame deformation (the center column) change their
  - local mesh topologies (per-vertex connectivities), and
  - global mesh topologies (mesh models of t are genus-0 and those of $t+1$ and $t+2$ are genus-1).

  On the other hand, the results of the inter-frame deformation (the right column) preserves both global and local mesh topologies while they represent the object with time-varying topology.

5.7 Summary

In this chapter, we improved our deformable mesh model to be able to cope with the object with time-varying global topology. The heterogeneous deformation model in Chapter 4 utilized a proximity-based force generation model. However, it cannot adapt to a complexity caused by the global object motion since it causes global mesh collisions which cannot deal with proximity-based approach.

To solve this problem, we introduced repulsive force to avoid global collision caused by the global complex object motion. This repulsive force and the silhouette preserving force enables the deformable mesh model to deform so as to represent the object shape with another global topology. That is, the deformable mesh model can "touch" itself.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we proposed a framework to estimate 3D shape and motion simultaneously from multi-viewpoint video. To realize simultaneous shape and motion estimation, we introduced deformable mesh model and two types of deformation process – intra-frame and inter-frame deformation. The intra-frame deformation estimates static 3D object shape from multi-viewpoint images at a frame. On the other hand, the inter-frame deformation estimates the dynamic object shape, i.e., the object shape and motion between two frames.

Our deformable mesh model can

- integrate several cues for accuracy, and
- realize shape continuity for stability.

Here, integration of multiple cues has following meanings.

Least commitment principle If one cue suggests an optimal state, we cannot determine whether it is actually an optima or a fake cause by noise. In this case, we have no choice but to determine based on a certain threshold if we use the stereo or other single-cue algorithms. However, with active contour model, such an uncertain cue can avoid decision (i.e., thresholding) until others be in consensus.

Mutual compensation Each cues have own advantages and disadvantages. For example, silhouette-based volume intersection method is stable, but cannot reconstruct accurate 3D object shape; its output represents just the visual

hull of the object and concave portions of the object cannot be reconstructed. In contrast, texture-based stereo method can reconstruct any kind of visible surfaces, but it is difficult to obtain dense and accurate stereo matching. Active contour model can combine both cues. If an object surface has a prominent texture, its position is determined accurately by mean of the stereo method. Otherwise, its position is smoothly interpolated according to its neighbors whose positions are determined by texture or silhouette.

Detailed situation analysis Active contour model realize better integration rather than just mixing several evaluated values. It can adaptively control the weightings of each cues based on the analysis between cues reflecting the situation of the object shape and motion.

In the intra-frame deformation (Chapter 2), we utilized following three constraints:

- photometric constraint,
- silhouette constraint, and
- smoothness constraint.

They represent the object shape by frame-and-skin model as described in Section 2.3.4.

In the inter-frame deformation, we extended the frame-and-skin model which models the object shape to be the heterogeneous deformation model by adding constraints which model the object motion. First, we modeled the object motion as totally warping (Chapter 3). We estimated the object motion roughly from 1) observed object silhouettes at two frames, and 2) motion history of the object. This is an example of mutual compensation between cues (Section 3.4). Next, we modeled the object motion as mixture of warping and rigid motion (Chapter 4). To find rigid regions, we clustered roughly estimated motion flow (Section 4.3). Finally, we introduced new force which is based on not geodesic but Euclidean proximity to cope with time-varying global topology (Chapter 5). This is because we implicitly assumed that a positive correlation between geodesic and Euclidean distance, and we defined forces so that they care the vertices only in their geodesic vicinity, but if the object global topology has change, this assumption breaks down.

6.2 Future Work

In this thesis we proposed intra- and inter-frame deformation, and with these two deformations, we presented a framework to estimate 3D shape and motion of the object:

Step 1. Estimate the object shape at frame t by the intra-frame deformation. We denote this mesh model by M_t and this is the first key frame.

Step 2. Estimate the object shape and motion at frame $t+1$ by the inter-frame deformation. Here, we use M_t as the initial shape, and obtain M_{t+1}.

Step 3. Repeat Step 2. and obtain M_{t+2} from M_{t+1}, M_{t+3} from M_{t+2}, and so on.

Step 4. Suppose the result of the inter-frame deformation $M_{t'}$ at frame t' cannot achieve a user-defined shape quality. In this case, estimate $M_{t'}$ by the intra-frame deformation, then go back to Step 2. and estimate $M_{t'+1}$ from $M_{t'}$.

As the result of this process, we can obtain 1) the shape and motion of the object and 2) physical and photometric properties of vertices. So similarly to the 2D video compression, we can expect that two-pass estimation is promising for more accuracy.

Next, the proposed method uses the 2D silhouette of the object as its input data. This means that the accuracy of estimated shape and motion depends on the accuracy of 2D silhouette. We assumed that it is accurate enough in this thesis, but it is well known that silhouette extraction of the object is not so easy problem. To solve this problem, we can use 3D shape estimation process itself. That is, if we can estimate accurate 3D shape, its projection onto each viewpoint should generate accurate 2D object silhouette. In other words, there is a projection/back-projection constraint between 3D shape and 2D silhouettes. While visual cone intersection method utilizes only the boundaries of 2D silhouettes and generates 3D shape (visual hull) straightforward, we can develop a method which estimates consensus 3D shape and 2D silhouettes simultaneously.

Bibliography

[BD00] Eugene Borovikov and Larry Davis. A distributed system for real-time volume reconstruction. In *Proceedings of the International Workshop on Computer Architectures for Machine Perception*, pages 183–189, Padova, Italy, September 2000.

[BG93] Jean Daniel Boissonnat and Bernhard Geiger. Three dimensional reconstruction of complex shapes based on the delaunay triangulation. Technical report, INRIA Technical Report RR-1697, 1993.

[BL01] Andrea Bottino and Aldo Laurentini. A silhouette-based technique for the reconstruction of human movement. *Computer Vision and Image Understanding*, 83:79–95, 2001.

[BLS03] Jörg Bredno, Thomas M. Lehmann, and Klaus Spitzer. A general discrete contour model in two, three, and four dimensions for topology-adaptive multichannel segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):550–563, 2003.

[BSM+03] Hector M. Briceño, Pedro V. Sander, Leonard McMillan, Steven Gortler, and Hugues Hoppe. Geometry videos: A new representation for 3d animations. In *Proceedings of the Annual Conference on Computer Graphics (SIGGRAPH)*, pages 136–146, 2003.

[Bur81] D.J. Burr. A dynamic model for image registration. *Computer Graphics and Image Processing*, 15:102–112, 1981.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the Annual Conference on Computer Graphics (SIGGRAPH)*, pages 43–54, 1998.

[CKBH00] Kong Man Cheung, Takeo Kanade, Jean-Yves Bouguet, and Mark Holler. A real time system for robust 3d voxel reconstruction of hu-

man motions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 714–720, South Carolina, USA, June 2000.

[CMN97] Jon Christensen, Joe Marks, and J. Thomas Ngo. Automatic motion synthesis for 3d mass-spring models. *The Visual Computer*, 13(1):20–28, 1997.

[CZ00] Geoffrey Cross and Andrew Zisserman. Surface reconstruction from multiple views using apparent contours and surface texture. In *Proceedings of NATO Advanced Research Workshop on Confluence of Computer Vision and Computer Graphics*, pages 25–47. Kluwer Academic Publishers, 2000.

[DM01] Hervé Delingette and Johan Montagnat. Shape and topology constraints on parametric active contours. *Computer Vision and Image Understanding*, 83(2):140–171, 2001.

[ES02] Carlos Hernández Esteban and Francis Schmitt. Multi-stereo 3d object reconstruction. In *Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission*, pages 159–167, Padova, Italy, June 2002.

[FL94] Pascal Fua and Yvan G. Leclerc. Using 3-dimensional meshes to combine image-based and geometry-based constraints. In *Proceedings of the European Conference on Computer Vision*, pages 281–291, 1994.

[FL95] Pascal Fua and Yvan G. Leclerc. Object-centered surface reconstruction: combining multi-image stereo and shading. *International Journal of Computer Vision*, 16(1):35–55, 1995.

[Fua95] Pascal Fua. Reconstructing complex surfaces from multiple stereo views. In *Proceedings of the International Conference on Computer Vision*, pages 1078–1085, Jun 1995.

[GM04] Bastian Goldlüecke and Marcus Magnor. Space-time isosurface evolution for temporally coherent 3d reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 350–355, Washington, D.C., USA, June 2004.

[HH96] Tony Heap and David Hogg. Towards 3d hand tracking using a deformable model. In *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition*, pages 140–145, 1996.

[Hor87] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America. A*, 4(4):629–642, Apr 1987.

[IR03] Lawrence Ibarria and Jarek Rossignac. Dynapack: space-time compression of the 3d animations of triangle meshes with fixed connectivity. In *Proceedings of the Annual Conference on Computer Graphics (SIGGRAPH)*, pages 126–135, 2003.

[IS02] John Isidoro and Stan Sclaroff. Stochastic mesh-based multiview reconstruction. In *Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission*, pages 568–577, Padova, Italy, July 2002.

[KKI99] Yukiko Kenmochi, Kazunori Kotani, and Atsushi Imiya. Marching cubes method with connectivity. In *Proceedings of 1999 International Conference on Image Processing*, pages 361–365, Kobe, Japan, oct 1999.

[KRN97] Takeo Kanade, Peter Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Multimedia*, pages 34–47, 1997.

[KS99] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. In *Proceedings of the International Conference on Computer Vision*, pages 307–314, Kerkyra, Greece, September 1999.

[KWT88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[Lau95] Aldo Laurentini. How far 3d shapes can be understood from 2d silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):188–195, 1995.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the*

Annual Conference on Computer Graphics (SIGGRAPH), pages 163–169. ACM Press, 1987.

[MTG97] Saied Moezzi, Li-Cheng Tai, and Philippe Gerard. Virtual view generation for 3d digital video. *IEEE Multimedia*, pages 18–26, 1997.

[MWTN04] Takashi Matsuyama, Xiaojun Wu, Takeshi Takai, and Shohei Nobuhara. Real-time 3d shape reconstruction, dynamic 3d mesh deformation and high fidelity visualization for 3d video. *Computer Vision and Image Understanding*, 96:393–434, December 2004.

[OK98] Naoya Ohta and Kenichi Kanatani. Optimal estimation of three-dimensional rotation and reliability evaluation. In *Proceedings of the European Conference on Computer Vision*, pages 175–187, Germany, Jun 1998.

[PF03] Ralf Plänkers and Pascal Fua. Articulated soft objects for multiview shape and motion capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1182–1187, 2003.

[Pro95] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In Wayne A. Davis and Przemyslaw Prusinkiewicz, editors, *Graphics Interface '95*, pages 147–154. Canadian Human-Computer Communications Society, 1995.

[SD99] Steven Seitz and Charles Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 25(3), November 1999.

[SF95] W. Brent Seales and Olivier D. Faugeras. Building three-dimensional object models from image sequences. *Computer Vision and Image Understanding*, 61(3):308–324, 1995.

[SHIR95] Heung-Yeung Shum, Martial Hebert, Katsushi Ikeuchi, and Raj Reddy. An integral approach to free-formed object modeling. In *Proceedings of the International Conference on Computer Vision*, pages 870–875, Jun 1995.

[VBR+99] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proceedings of the In-*

ternational Conference on Computer Vision, volume 2, pages 722 – 729, September 1999.

[VBSK00] Sundar Vedula, Simon Baker, Steven Seitz, and Takeo Kanade. Shape and motion carving in 6d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2000.

[WC01] Kwan-Yee Kenneth Wong and Roberto Cipolla. Structure and motion from silhouettes. In *Proceedings of the International Conference on Computer Vision*, volume II, pages 217–222, Vancouver, Canada, July 2001.

[WI95] Mark D. Wheeler and Katsushi Ikeuchi. Iterative estimation of rotation and translation using the quaternion. Technical Report CMU-CS-95-215, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1995.

[WSI98] Mark D. Wheeler, Yoichi Sato, and Katsushi Ikeuchi. Consensus surfaces for modeling 3d objects from multiple range images. In *Proceedings of the International Conference on Computer Vision*, page 917. IEEE Computer Society, 1998.

[WUM95] Toshikazu Wada, Hiroyuki Ukida, and Takashi Matsuyama. Shape from shading with interreflections under proximal light source, -3d shape reconstruction of unfolded book surface from a scanner image-. In *Proceedings of the International Conference on Computer Vision*, pages 66–71, Jul 1995.

[WWTM00] Toshikazu Wada, Xiaojun Wu, Shogo Tokai, and Takashi Matsuyama. Homography based parallel volume intersection: Toward real-time reconstruction using active camera. In *Proceedings of the International Workshop on Computer Architectures for Machine Perception*, pages 331–339, Padova, Italy, September 2000.

[XP98] Chenyang Xu and Jerry L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369, Mar 1998.

[YS03a] Anthony J. Yezzi and Stefano Soatto. Deformation: Deforming motion, shape average and the joint registration and approximation

of structures in images. *International Journal of Computer Vision*, 53(2):153–167, 2003.

[YS03b] Anthony J. Yezzi and Stefano Soatto. Stereoscopic segmentation. *International Journal of Computer Vision*, 53(1):31–43, 2003.

List of Publications

Journal Paper

- Takashi Matsuyama, Xiaojun Wu, Takeshi Takai, and Shohei Nobuhara. Real-time 3d shape reconstruction, dynamic 3d mesh deformation and high fidelity visualization for 3d video. *Computer Vision and Image Understanding*, 96:393–434, December 2004.

- Takashi Matsuyama, Takesi Takai, Xiaojun Wu, and Shohei Nobuhara. Generation, editing, and visualization of 3d video. *The Transactions of The Virtual Reality Society of Japan*, 7(4):521–532, Dec 2002.

- Shohei Nobuhara, Toshikazu Wada, and Takashi Matsuyama. 3d shape from multi-viewpoint images using deformable mesh model. *IPSJ Transactions on Computer Vision and Image Media*, 43:53–63, Dec 2002.

International Conference

- Shohei Nobuhara and Takashi Matsuyama. Heterogeneous deformation model for 3d shape and motion recovery from multi-viewpoint images. In *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 566–573, Thessaloniki, Greece, September 2004.

- Shohei Nobuhara and Takashi Matsuyama. Dynamic 3d shape from multi-viewpoint images using deformable mesh models. In *Processings of 3rd International Symposium on Image and Signal Processing and Analysis*, pages 192–197, Rome, Italy, September 2003.

- Takashi Matsuyama, Xiaojun Wu, Takeshi Takai, and Shohei Nobuhara. Real-time generation and high fidelity visualization of 3d video. In *Processings of Mirage 2003*, pages 1–10, 2003 (Invited paper).

Presentation

- Shohei Nobuhara, Toshikazu Wada, and Takashi Matsuyama. 3d shape from multi-viewpoint images using deformable mesh model. In *Processings of the IPSJ Meeting on Image Recognition and Understanding (MIRU)*, pages 247–254, Jul 2002.

- Xiaojun Wu, Shohei Nobuhara, Toshikazu Wada, and Takashi Matsuyama. Real-time 3d shape reconstruction and refinement from multi-viewpoint image sequences. SIGNotes CVIM No.131-009, Information Processing Society of Japan, 1 2002.